BRIEFING SESSION ON NSS ICT ELECTIVE – Software Development

Centre for Information Technology in Education, HKU

Elective Option D: Software Development

- Topic A on “Programming” (47 hours)
  - Choose a programming language from Pascal, C, Visual Basic, Java
- Topic B on “Programming Languages” (12 hours)
  - Programming paradigms
  - Language translators and compilers

Elective Option D: Software Development

Topic	Learning Outcomes	Remarks
a. Programming (47 hours)		
i. Problem definition and analysis	• Define proble • Identify the in in solving a pr • Plan solutions appropriate pr • Describe the c refinement. • Apply structu program writi	
ii. Design of solution	• Select appropr solution. • Represent alg block diagram	

Topic	Learning Outcomes	Remarks
	• Design sets of test data for program testing.	Test data discussed
v. Documentation	• Develop the habit of documenting the processes of program development.	
	• Recognise various documents for documenting a program.	The docu represent test data a
b. Programming Languages (12 hours)		
i. Programming paradigms	• Be aware of the evolution of programming languages.	
	• Recognise the programming paradigms involved in procedural, logic, object-oriented and query languages.	One progr paradigm
	• Describe the criteria for selecting a programming language for a specific problem.	
ii. Language translators and compilers	• Define code generation, linkers and	

Topic A

Programming

Topic A: Programming

A1	Getting Familiar with the Programming Environment	3 periods	2 hours
A2	Input and Output	3 periods	2 hours
A3	Arithmetic Operations and Functions	4½ periods	3 hours
A4	Selection Structure	6 periods	4 hours
A5	Iteration Structure	7½ periods	5 hours
A6	Modular Programming	9 periods	6 hours
A7	Arrays	6 periods	4 hours
A8	Strings	4½ periods	3 hours
A9	File Handling	6 periods	4 hours
A10	Searching, Sorting and Merging	10½ periods	7 hours
A11	Stacks, Queues and Linked Lists	10½ periods	7 hours
	Total:	70½ periods	47 hours

* 1 period = 40 minutes

Resource package for Topic A

- 11 Chapters (A1 – A11)
- 4 Learning tasks
  - A1 – A3: Learning Task A #1
  - A4 – A6: Learning Task A #2
  - A7 – A9: Learning Task A #3
  - A10 – A11: Learning Task A #4
- 4 Assessment tasks
  - A1 – A3: Assessment Task A #1
  - A4 – A6: Assessment Task A #2
  - A7 – A9: Assessment Task A #3
  - A10 – A11: Assessment Task A #4
- Teaching plans for each chapter
- Source programs of examples and practical tasks
- List of references

Resource package for Topic B

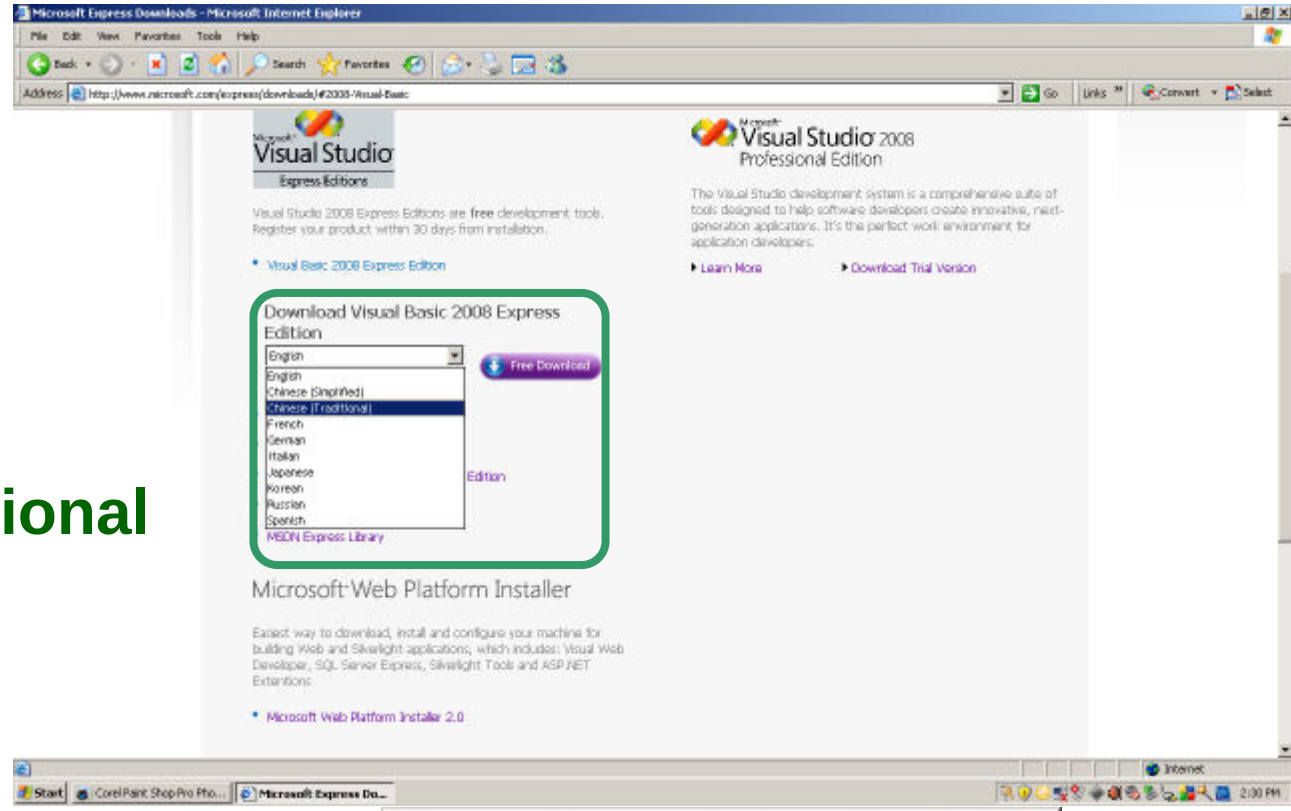
- 2 Chapters (B1 – B2)
- 2 Learning tasks
- 2 Assessment tasks
- Teaching plans for each chapter
- List of references

Highlight on Visual Basic

- Pascal, C, Visual Basic, Java
- More resources available for Pascal and C
- Highlights on **Visual Basic** and **Java**

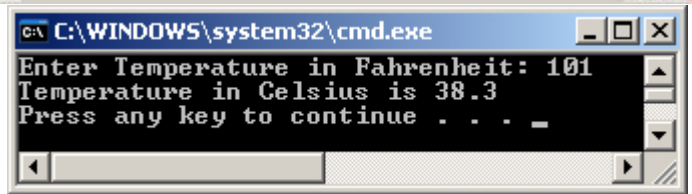
Highlight on Visual Basic

- Visual Basic 2008 Express
  - http://www.microsoft.com/express/Downloads/#2008-Visual-Basic



- **Free version**
- **English**
- **Chinese Traditional**

- Console application



Minimalist approach

- Students start as soon as possible on meaningful tasks
  - Understand the instructions with a not-so-steep learning curve
  - Use of VB Immediate Window

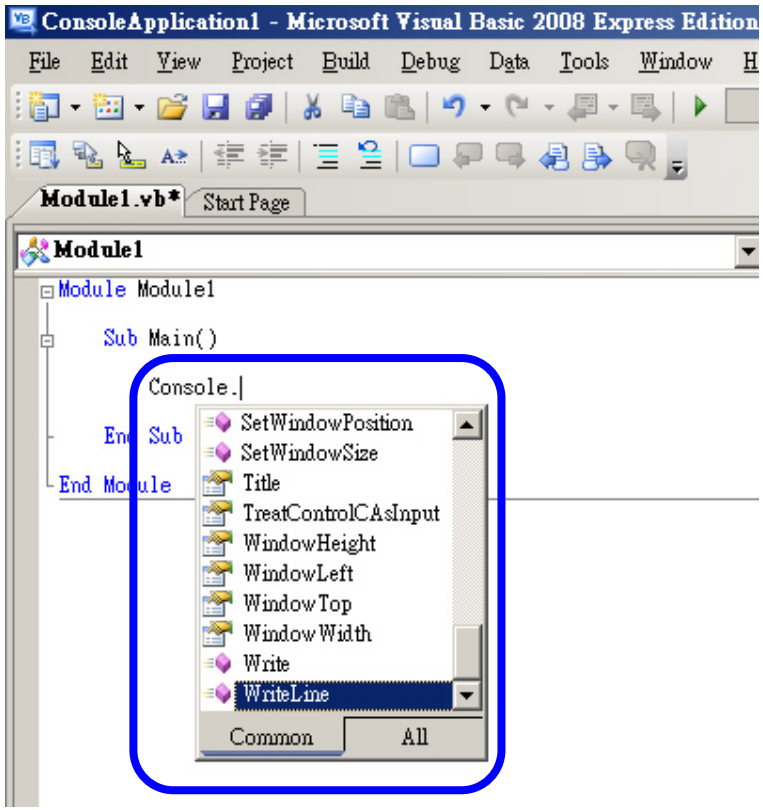
```
Immediate Window
length = 3
width = 4
? length * width
12 {Integer}
    Integer: 12 {Integer}
```

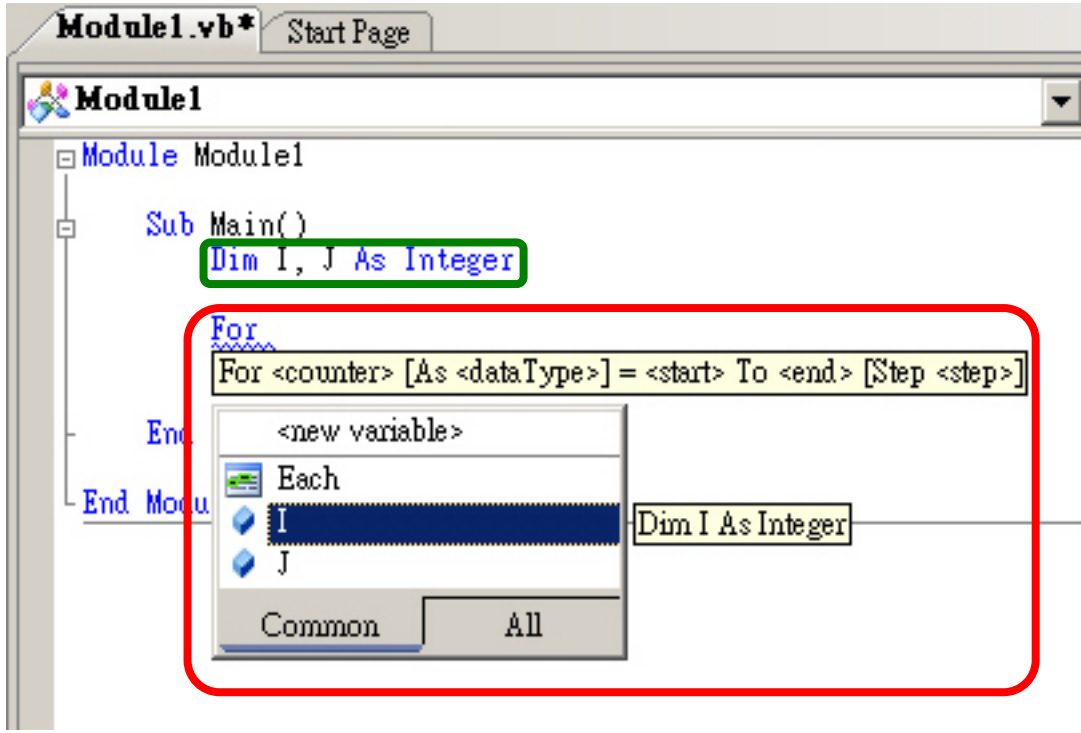
```
Immediate Window
alertmsg = "Wrong input!"
? alertmsg.Substring(8)
"put!" {String}
    String: "put!"
? alertmsg.Substring(6, 2)
"in" {String}
    String: "in"
```

```
Immediate Window
Randomize
? Int(Rnd()*6)+1
2.0
? Int(Rnd()*6)+1
1.0
? Int(Rnd()*6)+1
5.0
? Int(Rnd()*6)+1
4.0
? Int(Rnd()*6)+1
3.0
? Int(Rnd()*6)+1
3.0
? Int(Rnd()*6)+1
4.0
? Int(Rnd()*6)+1
2.0
? Int(Rnd()*6)+1
5.0
? Int(Rnd()*6)+1
1.0
? Int(Rnd()*6)+1
5.0
? Int(Rnd()*6)+1
6.0
```

Minimalist approach

- VB's IntelliSense and code snippets
  - VB's IntelliSense feature offers assistance by displaying possible code snippets
  - Less syntax and typing errors

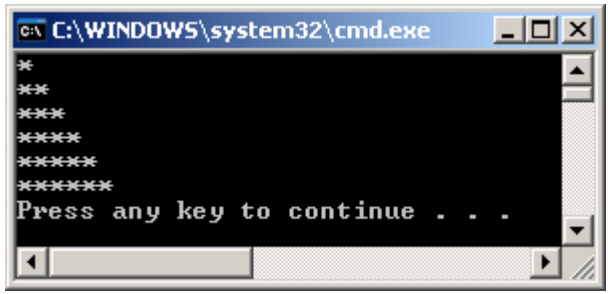




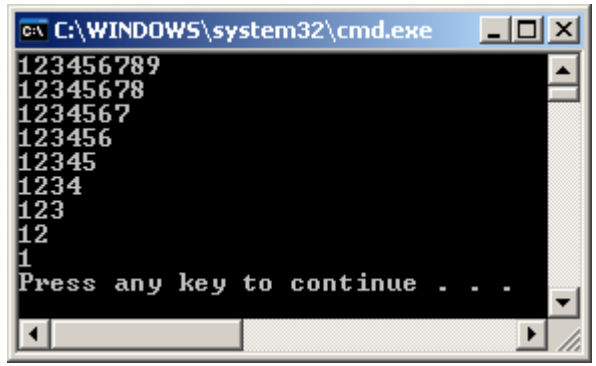
Minimalist approach

- Students start as soon as possible on meaningful tasks
  - Short and simple programs
  - Programs will be re-used for enhancement
- Examples are followed by similar practical tasks

- Example



- Practical Task



Minimalist approach

```
'This program converts temperature in Fahrenheit to Celsius
'and output the Celsius degree to 1 decimal place.

Module FtoC

    Sub Main()
        Dim Fahrenheit, Celsius As Single

        'Prompt the user to enter temperature in Fahrenheit
        Console.Write("Enter Temperature in Fahrenheit: ")
        Fahrenheit = Console.ReadLine()

        'Convert the temperature in Fahrenheit to Celsius
        Celsius = (Fahrenheit - 32) * 5 / 9

        'Output the temperature in Celsius to 1 decimal place
        Console.WriteLine("Temperature in Celsius is " & "{0:F1}", Celsius)

        'Check a condition whether the Celsius temperature is
        'greater than or equal to 36.8 and then output the message
        '"You have fever." accordingly
        If Celsius >= 36.8 Then
            Console.WriteLine("You have fever.")
        End If
    End Sub

End Module
```

- Examples are re-used and enhanced
  - Students are already familiar with the context and source code
  - Learn by variation

Dissection method

- Dissection is a pedagogical method to
  - highlight key features of the code
  - make a structured walk-through of the code
- Purposes of dissecting a program
  - explain newly encountered programming elements and technical terms in the code
  - explain the code in an easy-to-follow and step-by-step manner
- Each chapter is mostly led by dissecting examples

Dissection method

```
Module FtoC

    Sub Main()
        Dim Fahrenheit, Celsius As Single

        'Prompt the user to enter temperature in Fahrenheit
        Console.Write("Enter Temperature in Fahrenheit: ")
        Fahrenheit = Console.ReadLine()

        'Convert the temperature in Fahrenheit to Celsius
        Celsius = (Fahrenheit - 32) * 5 / 9

        'Output the temperature in Celsius to 1 decimal place
        Console.WriteLine("Temperature in Ce

        'Check a condition whether the Celsi
        'greater than or equal to 36.8 and t
        '"You have fever." accordingly.
        If Celsius >= 36.8 Then
            Console.WriteLine("You have feve
        End If
    End Sub

End Module
```

Two sample outputs are as follows:

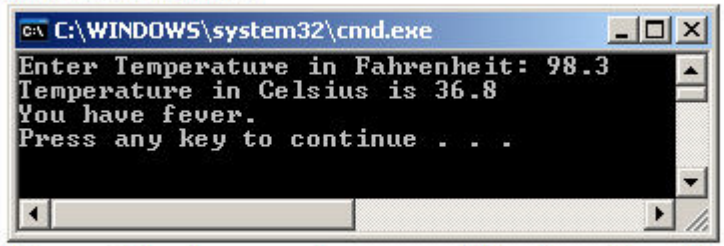


Fig. 1.2(a) The condition is met and a message "You have fever." is displayed.

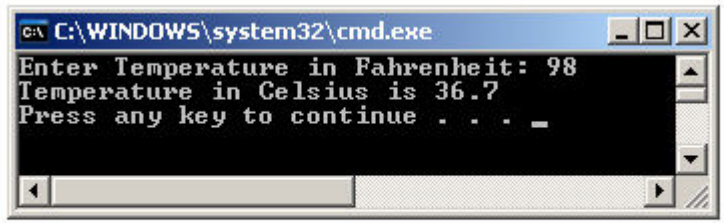


Fig. 1.2(b) The condition is ***not*** satisfied and the program stops after displaying the temperature in Celsius.

Dissection method

Lines 24 to 26 are the pieces of code added, apart from additional comments. The statement `If Celsius >= 36.8 Then` in line 24 is a statement to test whether the temperature in Celsius (stored in the variable `Celsius`) is greater than or equal to 36.8. The expression (`Celsius >= 36.8`) in the `If ... Then` statement is called a **condition**, which consists of comparing the value of a variable using a **relational operator**, >=, with a testing value 36.8. If the condition is met (or satisfied), i.e., the condition is evaluated as **true**, then the body in line 25 (`Console.WriteLine("You have fever.")`) in the `If ... Then` statement will be executed (**Fig. 1.2(a)**). The body is enclosed with the `If ... Then` statement and the `End If` statement, and `If` and `Then` are keywords of Visual Basic.

Good programming practice

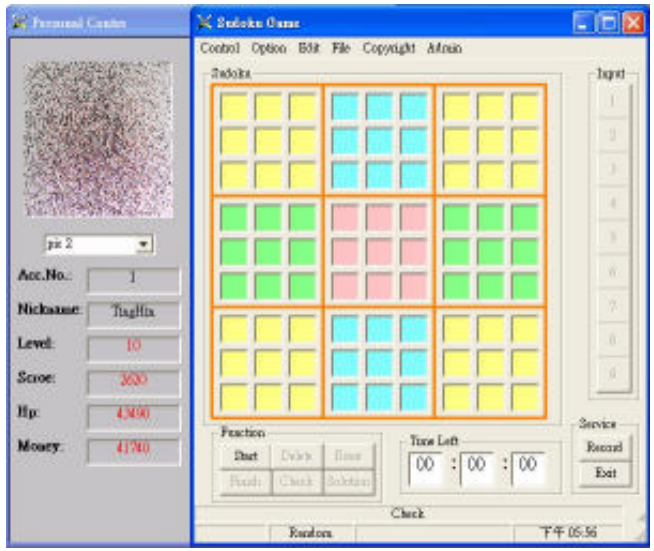
shown below:

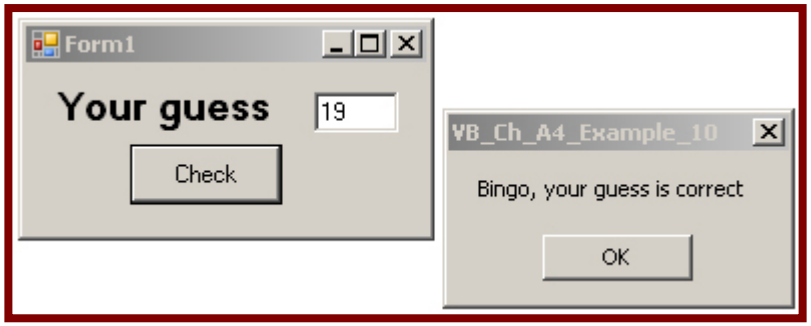
Good Programming Practice 2.6

Though numeric variables are initialised to zero by Visual Basic, to avoid confusion and to increase program readability, it is a good habit to initialise variables explicitly.

Windows Form Application

- Facilitate students to write Windows application for SBA project if they wish
- As appendix and it is optional






```
Public Class Form1
    Public HiddenNum As Integer

    Private Sub Form1_Load(ByVal sender As System.Objec
        Randomize()
        HiddenNum = Int(Rnd() * 100 + 1)

    End Sub

    Private Sub Button1_Click(ByVal sender As System.Ob
        Dim Guess As Integer

        Guess = Val(TextBox1.Text)
        If Guess < HiddenNum Then
            MsgBox("Your guess is too small")
        Else
            If Guess > HiddenNum Then
                MsgBox("Your guess is too large")
            Else
                MsgBox("Bingo, your guess is correct")
            End If
        End If
    End Sub
End Class
```

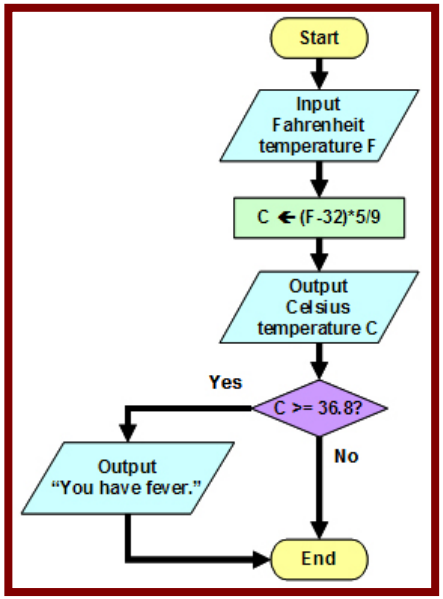
Overview

- Chap A1 – Getting Familiar with the Programming Environment
- Introduce Object-oriented concept
  - `Console.WriteLine("Hello world")`

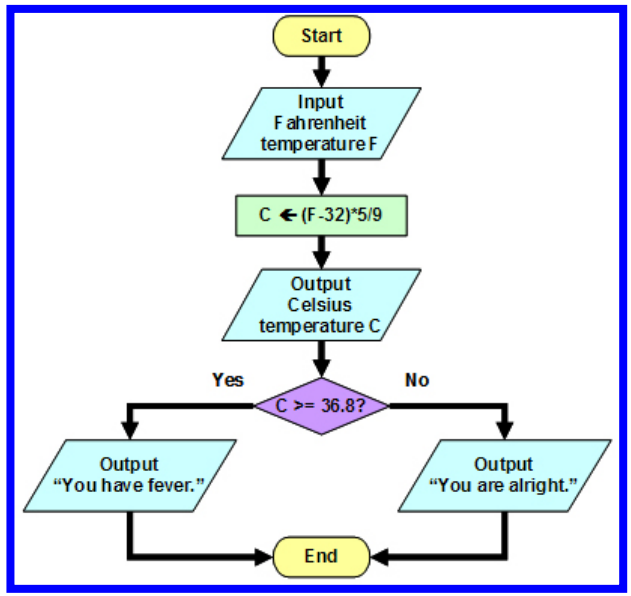
- Chap A2 – Input and Output
  - Input by prompting
- Chap A3 – Arithmetic operations and functions
  - Built-in functions (use of Immediate Windows)

Overview

- Chap A4 – Selection Structure

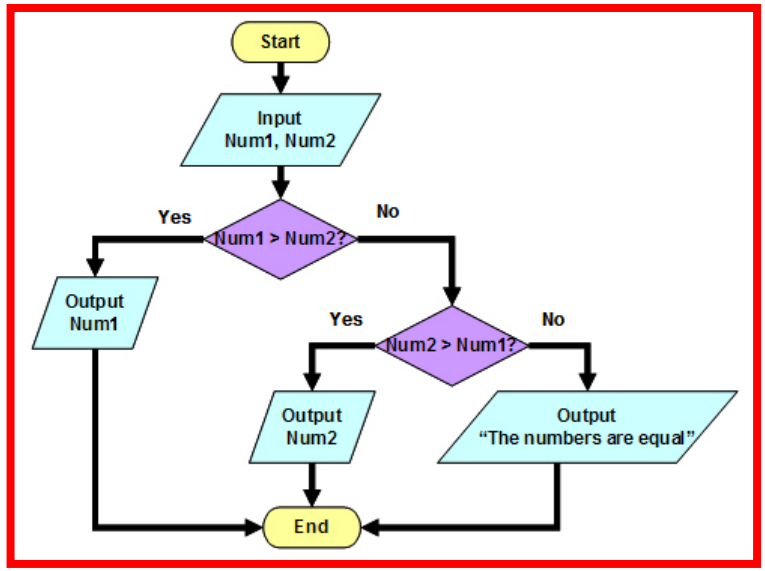


Single selection



Double selection

Nested selection



Overview

- Chap A5 – Iteration Structure
  - Counter, Accumulator, Flag
  - Data validation
  - Terminating value of the inner loop is controlled by the counter variable of the outer loop

Overview

- Chap A6 – Modular programming
  - Top-down approach, procedures
  - Local variables, global variables
  - Pass-by-value, pass-by-reference
  - User-defined functions
  - Step-wise refinement
  - Stub
  - Interpret errors
  - Debugging
  - Documentation

Overview

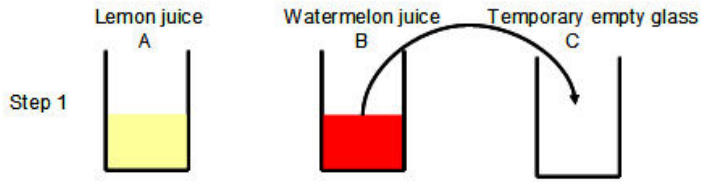
- “Recursion” is introduced in Chap A6 as an alternative solution to the same problem: evaluating factorial
- And it is adopted in merge sort in Chap A10

Overview

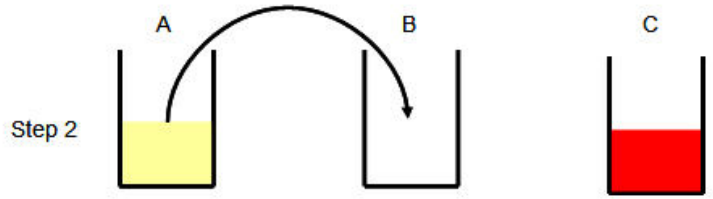
- Chap A7 – Arrays
  - Common array manipulations
  - Common pitfall: lower bound of arrays in Visual Basic starts from 0
  - User-defined data type: structure (to be used in Chap A11)
- Chap A8 – Strings
  - Use VB's Immediate Windows to try out many different string functions
- Chap A9 – File Handling
  - Fixed field size (use string functions in Chap A8)
- Chap A10 – Searching, Sorting and Merging
  - Linear search, Binary search
  - Swapping
  - Arrange 2 items in ascending/descending order
  - Bubble sort, Insertion sort, Merge sort
  - Emphasis on algorithm
  - Aware the existence of other efficient sorting algorithm, such as quick sort
  - Big O notation

Overview

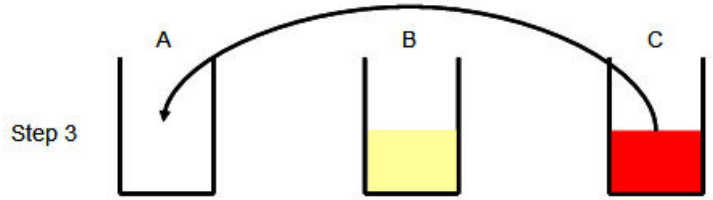
Swapping



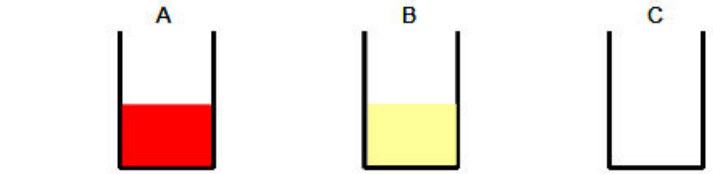
Step 1 is to pour watermelon juice from glass B to the empty glass C.



Once glass B is emptied, lemon juice can be poured from glass A to it.



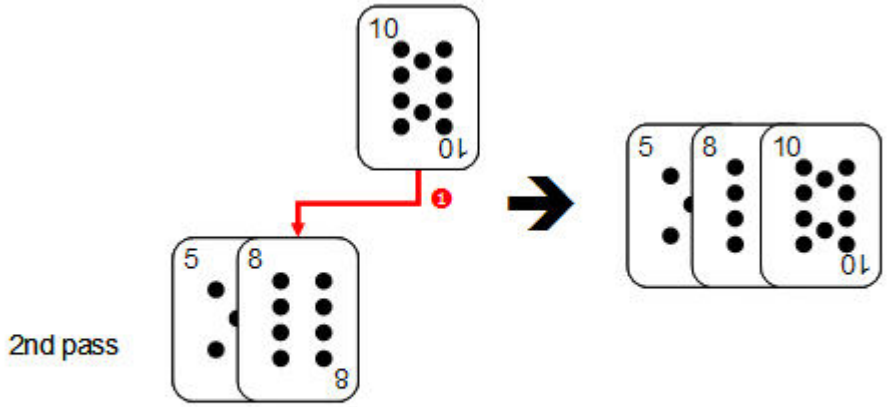
Glass A is empty after lemon juice is transferred to glass B. Watermelon juice in extra glass C can be poured to glass A.



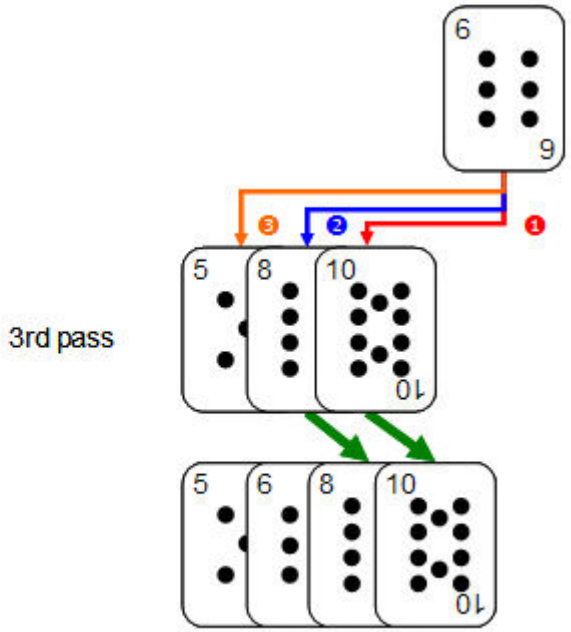
Lemon juice and watermelon juice are exchanged finally. The extra glass C is empty as it was, therefore, it serves as a temporary storage during the process of exchange.

Fig. 2.2 Exchange of juices requires an extra empty glass to hold a kind of juice temporarily.

Insertion sort



When the fourth card 6 is dealt, the comparisons made, from the right to the left, with the third, second and first card at hand show that it is to be inserted to the right of the first card and the original second and third card have to be moved to the right to make room for the fourth card to be inserted as the second card after insertion.



Overview

- Chap A11 – Stacks, Queues and Linked Lists
  - Implement in terms of arrays
  - No library function call

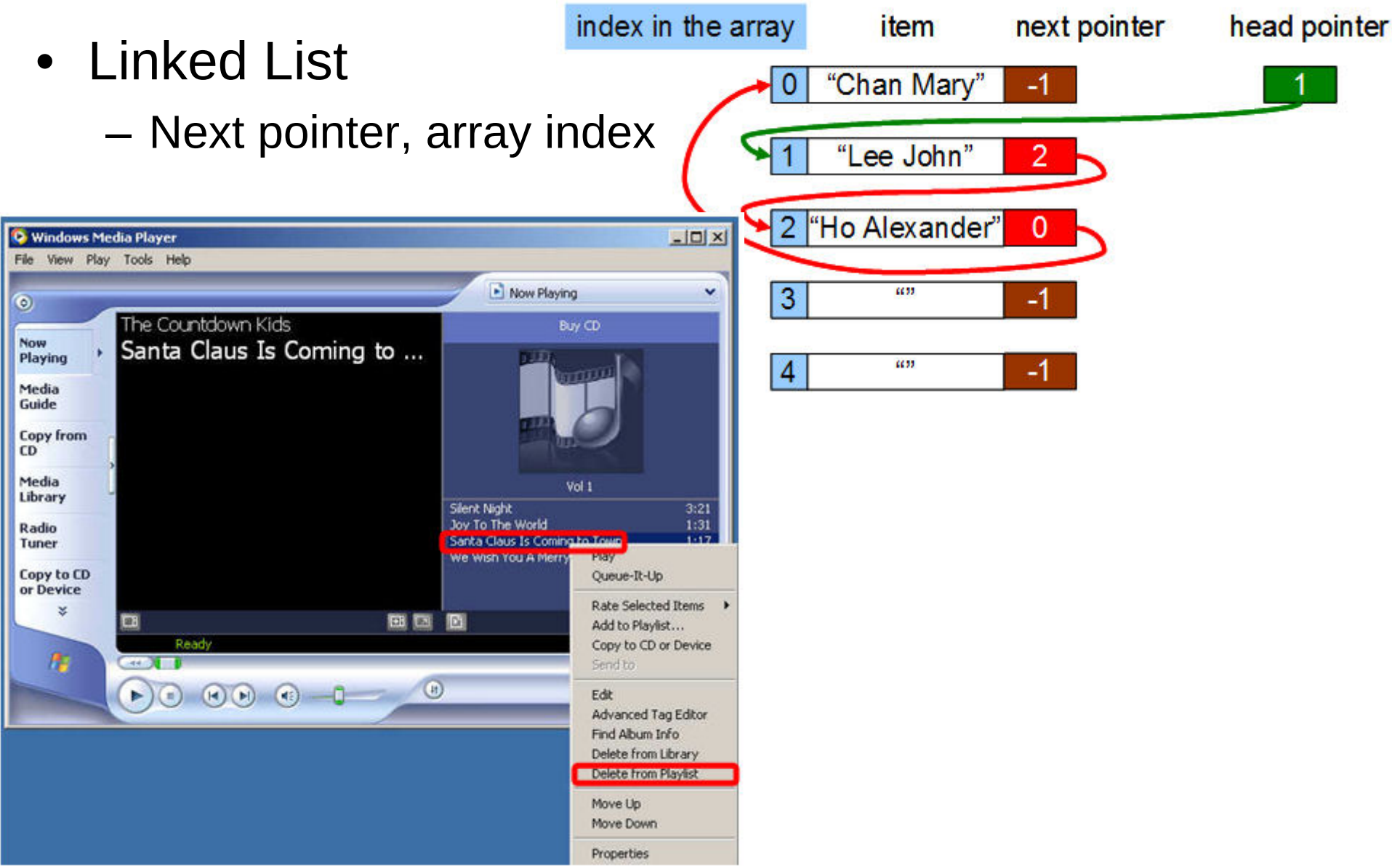
```
30      Sub Push(ByVal towel_colour As String, ByRef towel As stack_towel)
31          'upper bound of an array of a stack is stack_size - 1 because
32          'lower bound of an array in VB starts from 0
33          If towel.stack_pointer = stack_size - 1 Then
34              Console.WriteLine("The stack is full.")
35          Else
36              towel.stack_pointer += 1
37              towel.element(towel.stack_pointer) = towel_colour
38          End If
39      End Sub
```

Application of a Stack
– retrieve most recent items
(most recent calls from
Call register of a mobile phone)

- Linear queue and its drawbacks ➔ Circular queue

Overview

- Linked List
  - Next pointer, array index



Learning Task

- Consolidate students' learning after they have learnt a few chapters
- Go through a series of 4 learning tasks to complete a mini-project by implementing different features gradually
- The 4 tasks are NOT fragmented, but inter-related so that they can be combined to become a computer game
- Students gain experience to accomplish a project similar to a SBA project

Learning Task

Mastermind game

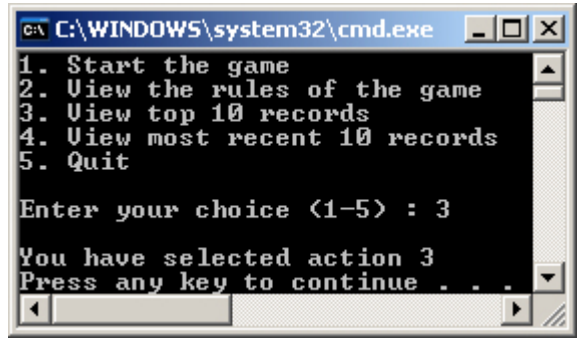
- Player tries to decode the secret code of 4 colour code pegs
- Codemaker gives feedback to the player for pegs of correct colour and in the correct position, and pegs of correct colour but in wrong position

Learning Task A #1

- Chap A1 – A3
  - Getting familiar with the programming environment
  - Input and Output
  - Arithmetic Operations and Functions

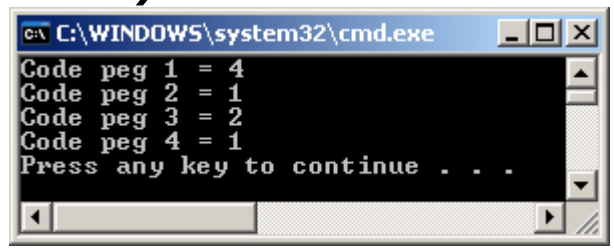
Learning Task A #1

- Design a menu (Chap A2)
  - Input by prompting



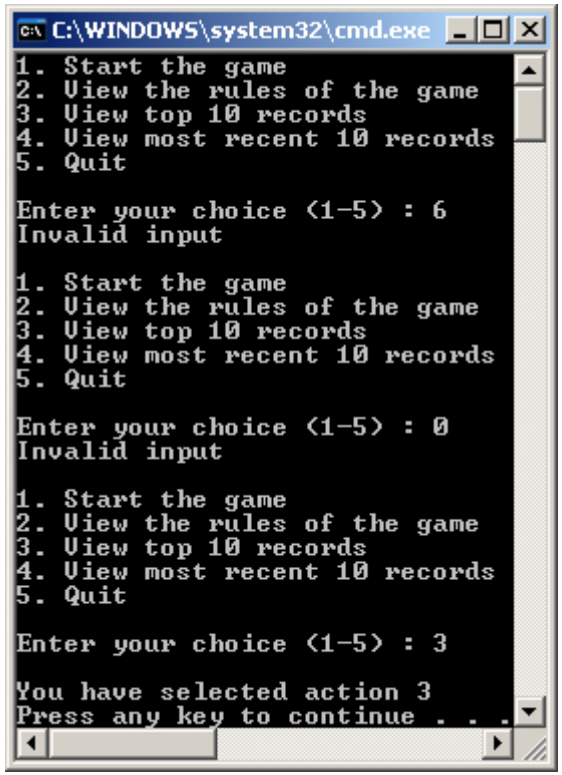
- Display in colour (Chap A1)
  - Learn by doing something unfamiliar, e.g.
    - `Console.ForegroundColor = ConsoleColor.Red`

- Generate secret code (Chap A3)
  - Generate random numbers



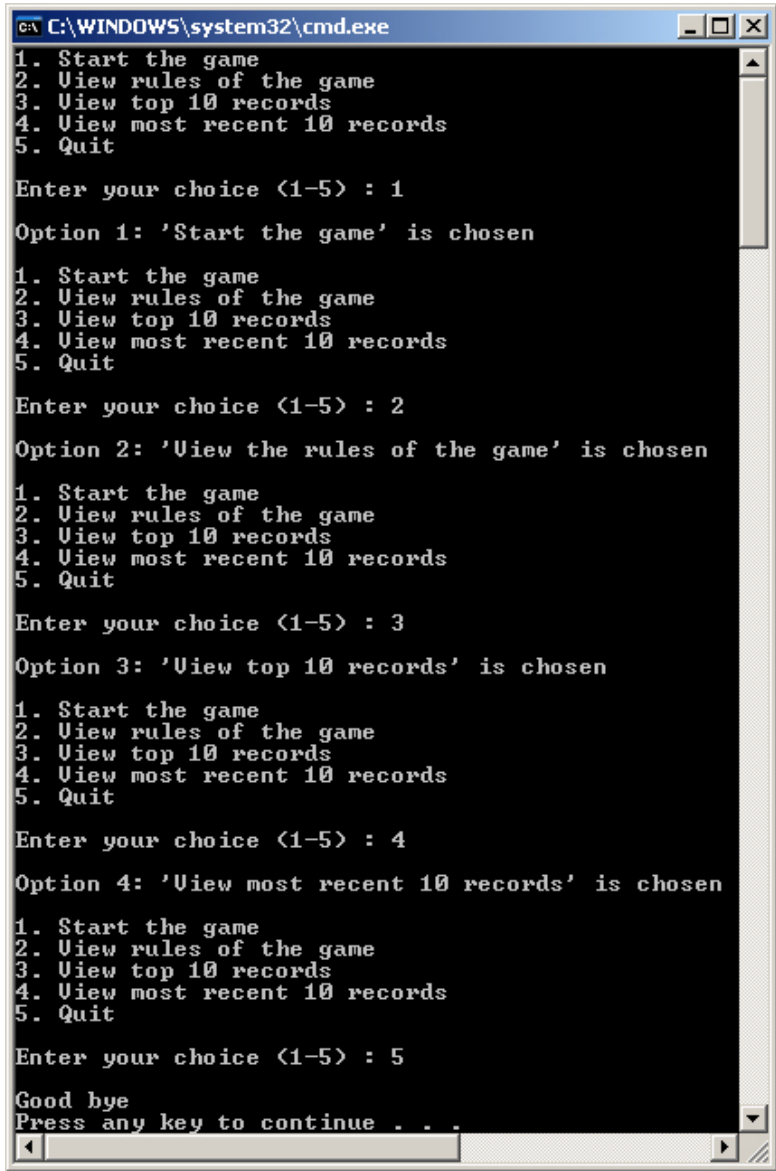
Learning Task A #2

- Validate player's choice (Chap A4, A5)
  - Flag
  - Selection Structure
  - Iteration Structure



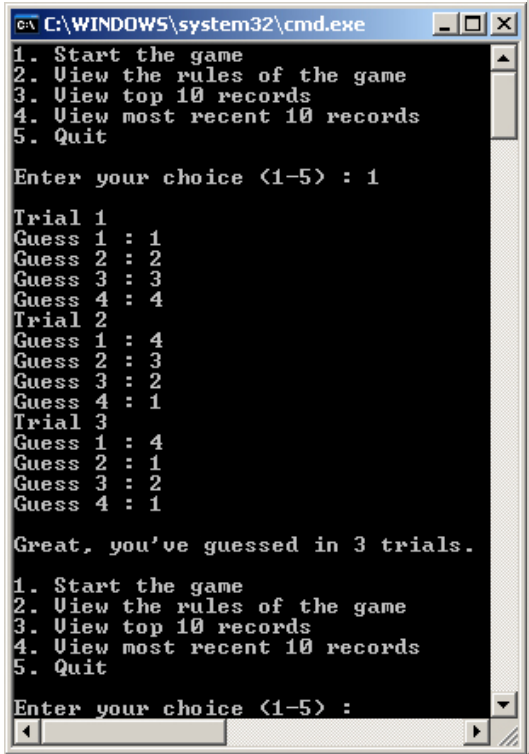
Learning Task A #2

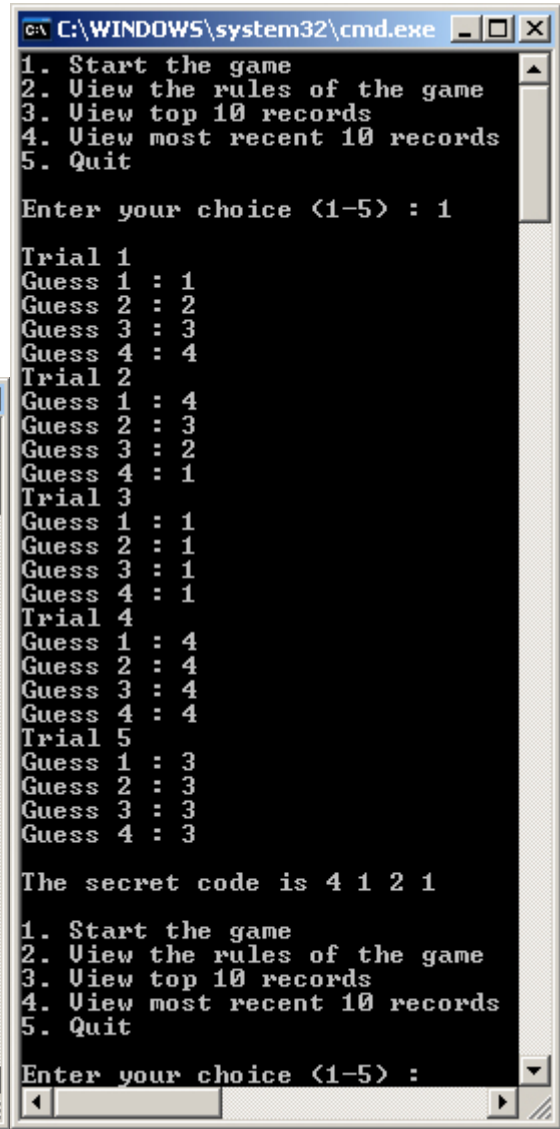
- Act according to player's choice (Chap A6)
  - Procedure (sub-routines)
  - Stub programming



Learning Task A #2

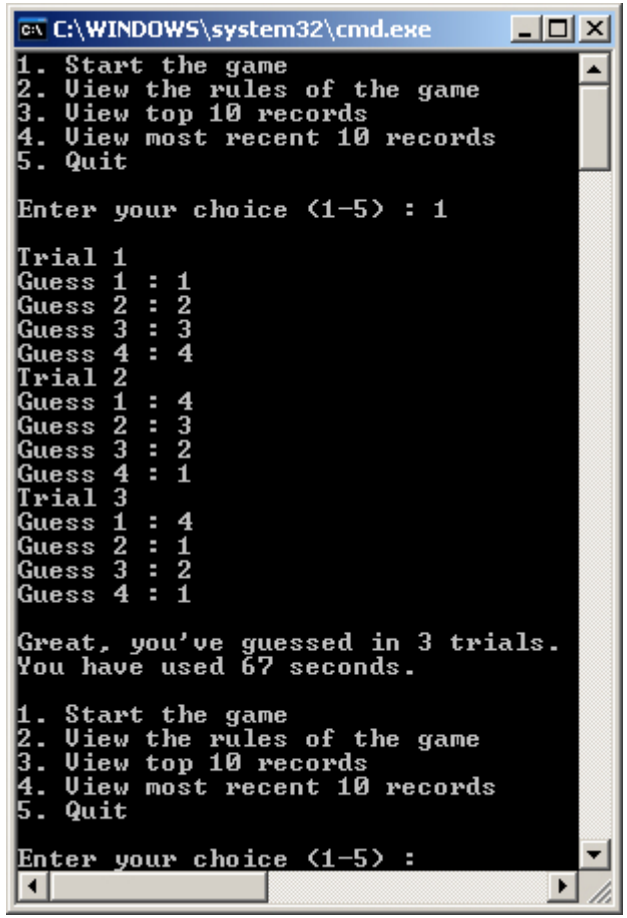
- Play a dummy game (Chap A4, A5, A6)
  - Counting
  - Selection Structure
  - Iteration Structure
  - Testing & Debugging





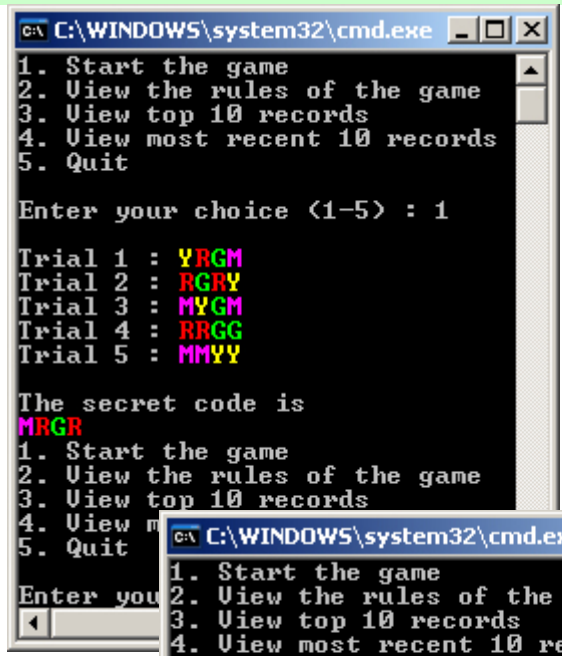
Learning Task A #2

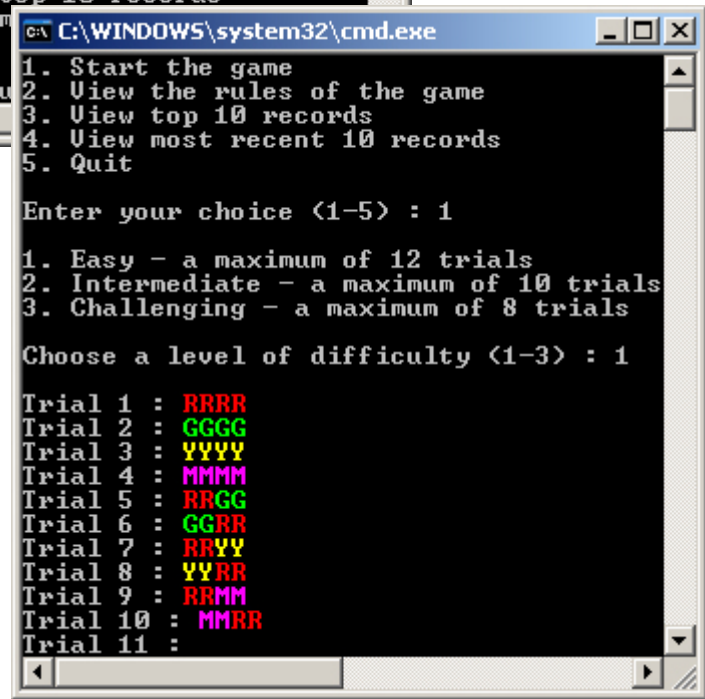
- Time the game
  - Learn by doing something unfamiliar, e.g. get the system time and store as a date data type



Learning Task A #3

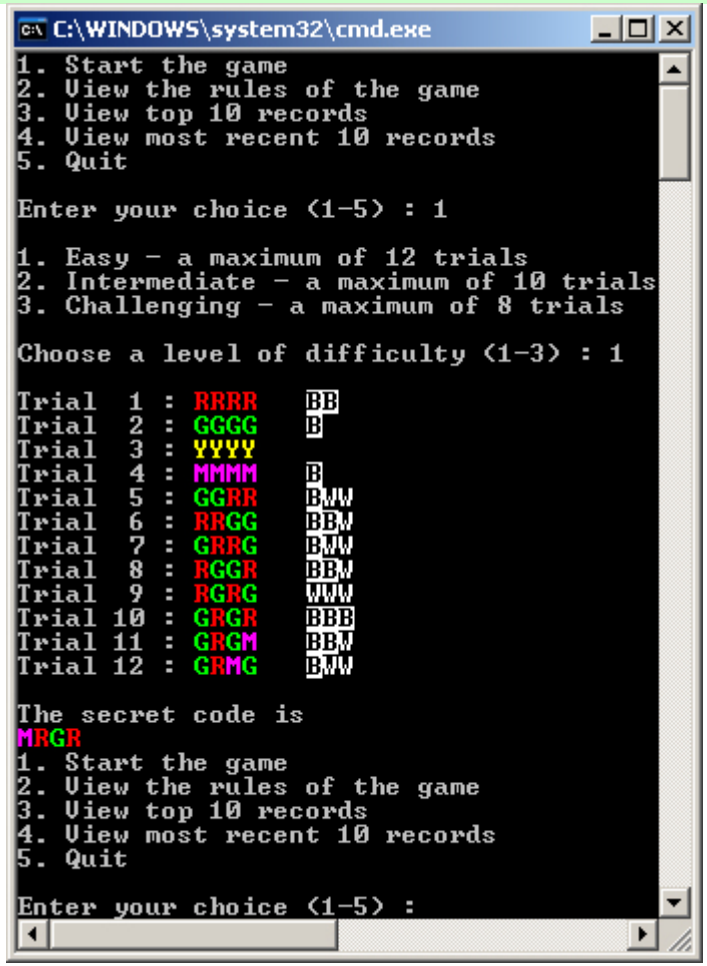
- Improve the data structure and user-interface (Chap A7, A8)
  - Array
  - Read single character from a keystroke
- Implement level of difficulty





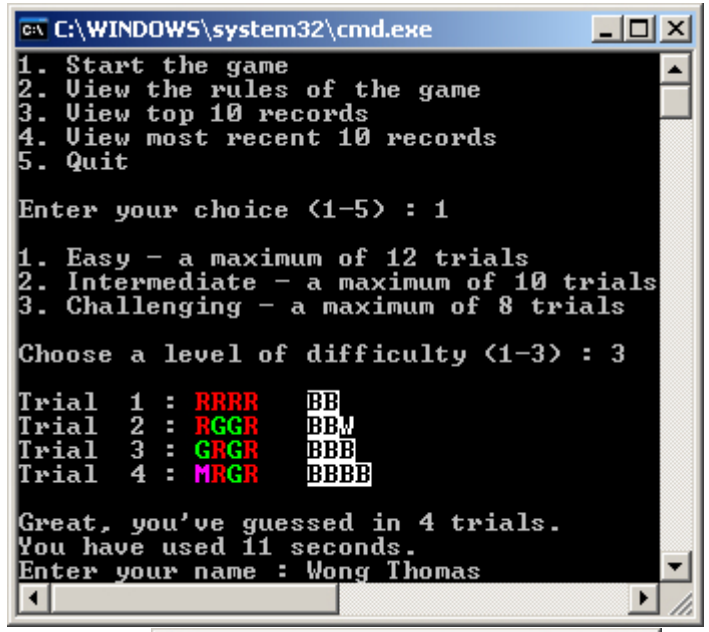
Learning Task A #3

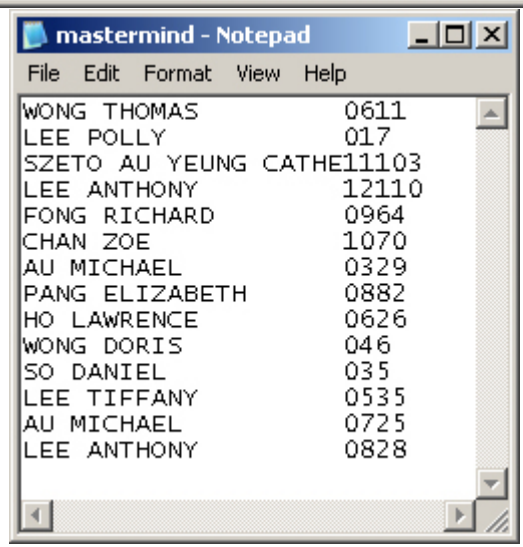
- Check the player's guess
  - Algorithm
    - ➢Count number of pegs of correct colour in the correct position (Black) first
    - ➢Count number of pegs of correct colour but in wrong position



Learning Task A #3

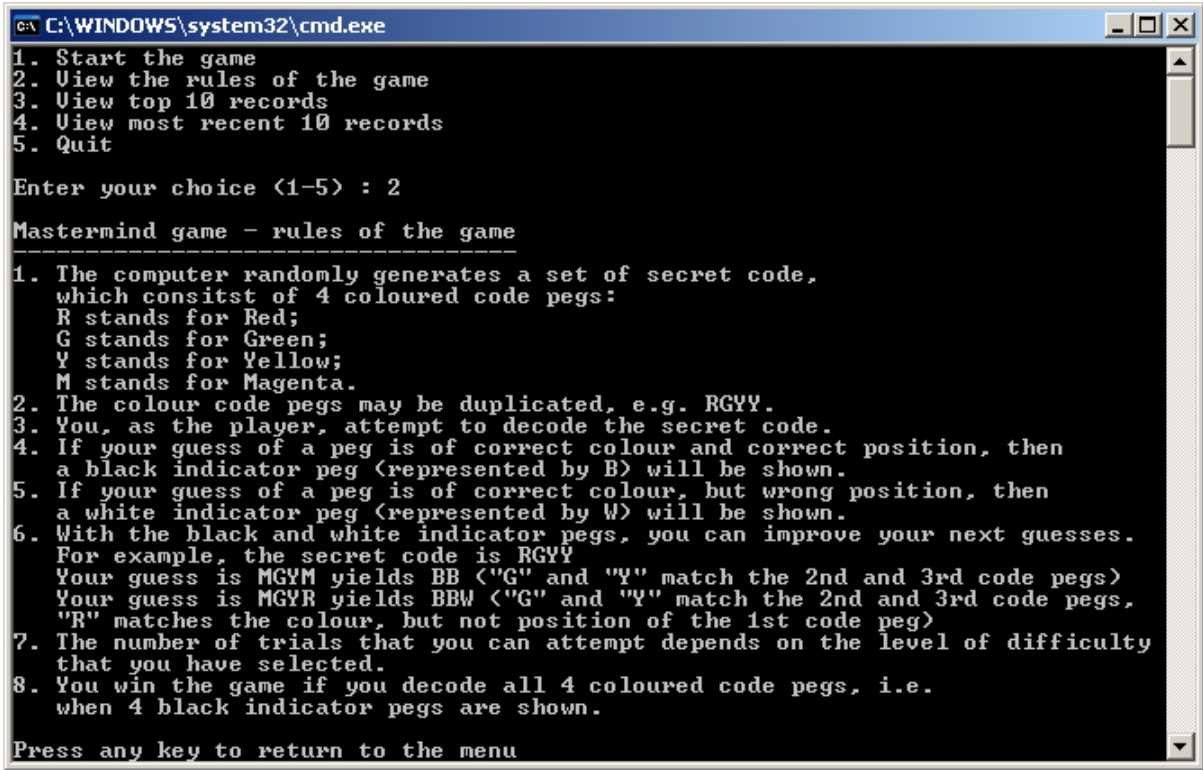
- Save winner's details to file (Chap A8, A9)
  - Strings
  - File Handling





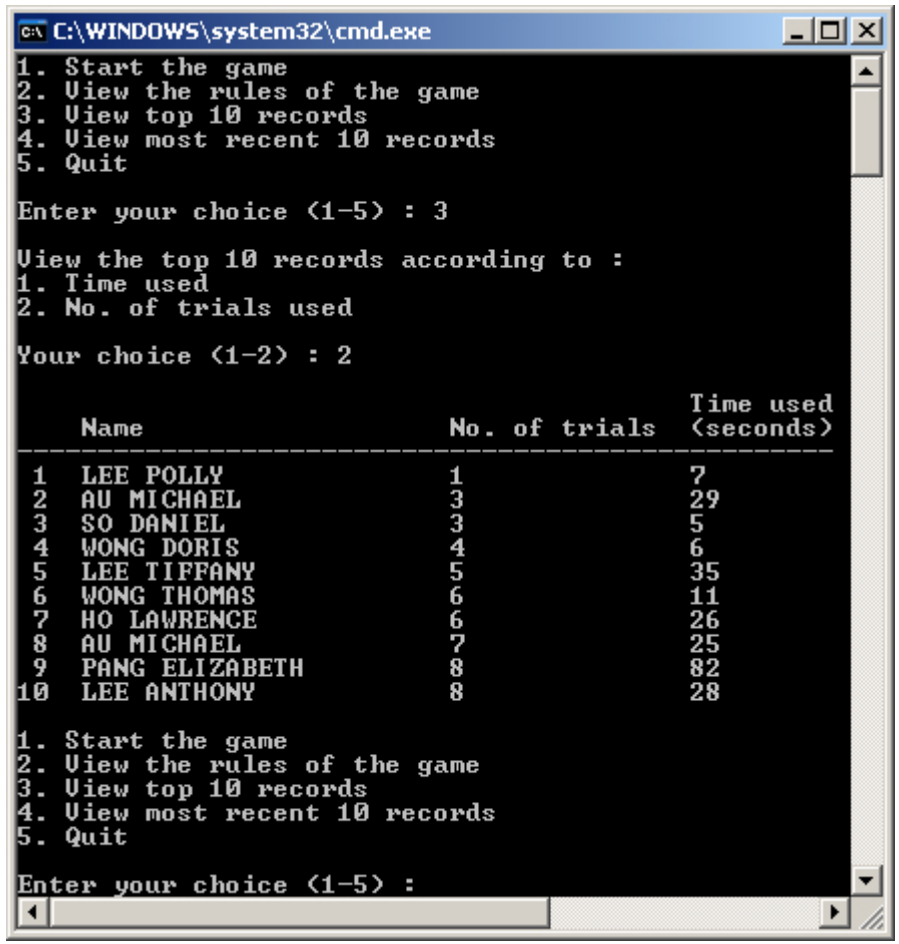
Learning Task A #3

- Read the rules of the game from file and display (Chap A9)
  - File Handling



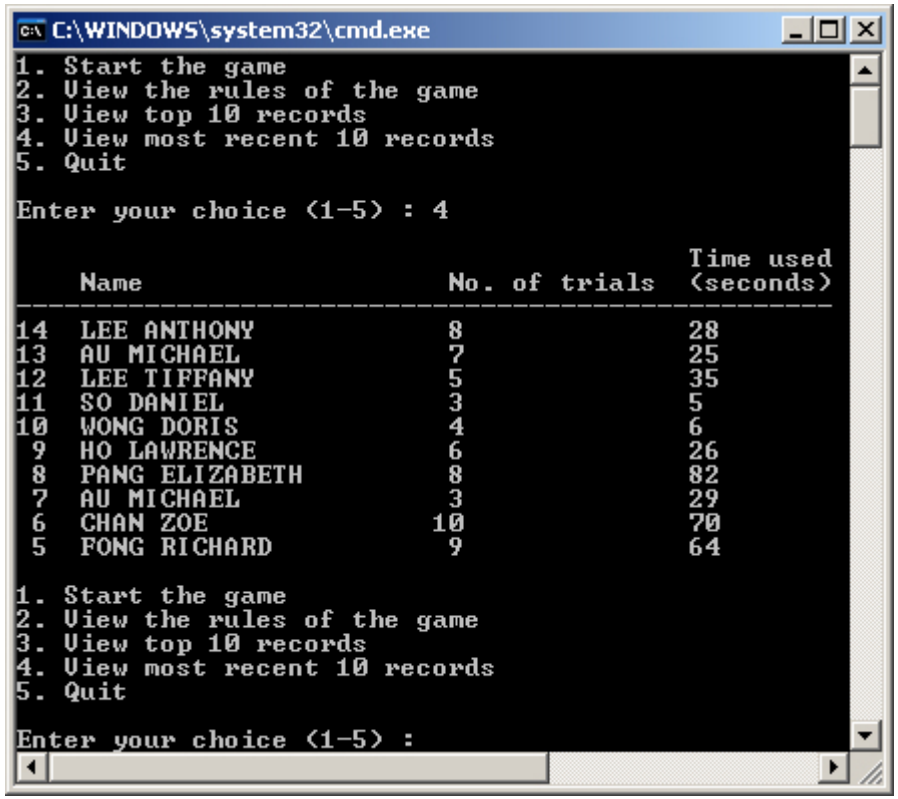
Learning Task A #4

- Sort the records of the winners (Chap A10)
  - Sorting



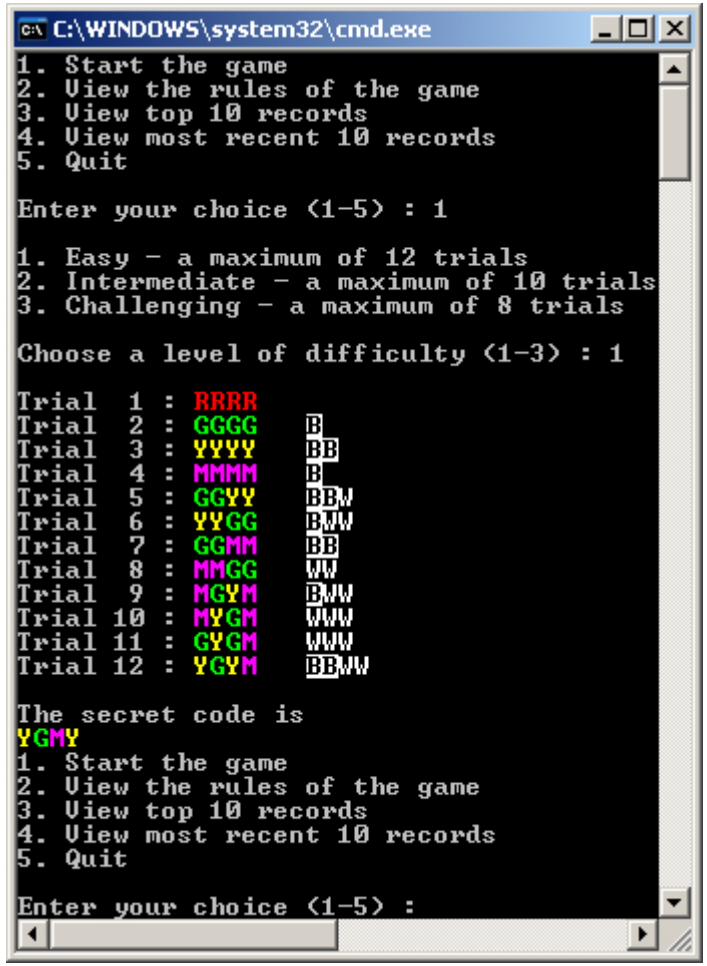
Learning Task A #4

- Retrieve most recent records (Chap A11)
  - Stacks



Learning Task A #4

- Finalize the game by removing the pre-set secret code



Assessment Task A #1 – #4

Chapter A1 **Getting familiar with the programming environment**
Chapter A2 **Input and output**
Chapter A3 **Arithmetic operations and functions**

Exercise / Quiz

1. Which of the following is a valid variable name in VB?
   - A. MyAge2010
   - B. 2010MyAge
   - C. _MyAge
   - D. My Age

```
8       Function Function2(ByVal x() As Integer, By
        As Integer) As Integer
9           Dim i, smallest_position As Integer
10          smallest_position = start_position
11          For i = start_position To x.GetUpperBo
12              If x(i) < [          ] Then
13                  [          ]
14              End If
15          Next
16          Return([          ])
17      End Function
```

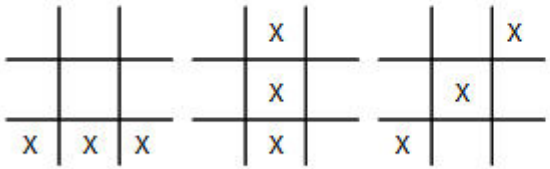
Line no.	Statement
1	`Function Encryption(ByVal x As String) As String`
2	`  Dim i As Integer`
3	`  Dim result As String`
4	`  result = ""`
5	`  For i = 0 To x.Length - 1`
6	`    result = result & Chr(Asc(x.Substring(i, 1)) + 3)`
7	`  Next`
8	`  Return(result)`
9	
10	`Sub Main`
11	`  Console.WriteLine(Encryption("gold4")`
12	`End Sub`

(a) Fill in the following table after the statement in line 6 is executed in each pass.

pass	Asc(x.Substring(i, 1))	Chr(SubString(i, 1)) + 3)	result
1			
2			
3			
4			
5			

Assessment Task A #1 – #4

12. Tic-Tac-Toe is a famous game for 2 players to take turn to mark crosses ("X") and noughts ("O") on spaces of a 3 by 3 grid. A player wins the game if he is the first one to mark 3 of his symbols either in a horizontal, vertical or diagonal row, e.g.



A one-dimensional array `grid(8)` is declared as `Char` type to represent spaces of the grid in the following way:

0	1	2
3	4	5
6	7	8

For instance, a cross is assigned to the centre space w

```
grid(4) = "X"
```

A nought is assigned to the lower right corner with th

```
grid(8) = "O"
```

(c) Write a user-defined function `Player_X_Win_Horizontal` to return a Boolean value `True` if the player who marks crosses wins by marking 3 crosses at one of the 3 horizontal rows. Otherwise, the function returns a Boolean value `False`. The array `grid` is passed to the function as a parameter.

```
Function Player_X_Win_Horizontal(ByVal g() As Char) As Boolean




End Function
```

Teaching plan for each chapter

Teaching plan for topic A: Programming

VB

Chapter A2 – Input and Output

Objectives

1. To understand how to use the methods `WriteLine` and `Write` for output and recognise their difference.
2. To use format specifier to display formatted output.
3. To declare constants and variables with appropriate data type.
4. To use assignment statements to assign values to a constant and a variable.
5. To understand how to use the method `ReadLine` for input.
6. To recognise input by prompting improves user-friendliness of a program.

Duration

2 hours in total (approximately equivalent to 3 periods, each period lasts 40 minutes)

Resources

1. Learning task sheet A #1 (The learning task should be attempted after completing chapter A3.)
2. Assessment task sheet A #1 (The assessment task should be attempted after completing chapter A3.)

Suggested time allocation

Learning outcomes	Duration in minutes	Section /Resources in the Package	Learning & Teaching activities	Teaching points / Explanatory notes
1. Identify the inputs and outputs involved in a solution.	40 min (1 period)	• Section 1	• Introduce the output method `Write` and recognise the difference between the methods	• Direct instruction. • Compare the effects of using the methods `Write` and `WriteLine`.

Topic B

Programming Languages

Topic B: Programming Languages

B1	Programming paradigms	10½ periods	7 hours
B2	Language translators and compilers	7½ periods	5 hours
	Total:	18 periods	12 hours

- Briefly in the current HKCEE CIT

- Selection of programming languages

HKCEE CIT

- Students should be aware that there exist different types of programming paradigm, such as imperative programming, object-oriented programming, logic programming, etc.
- Students should also be aware that different programming languages are designed to satisfy different specific purposes.

Topic B: Programming Languages

- But, it is required in details in the current HKAL CS

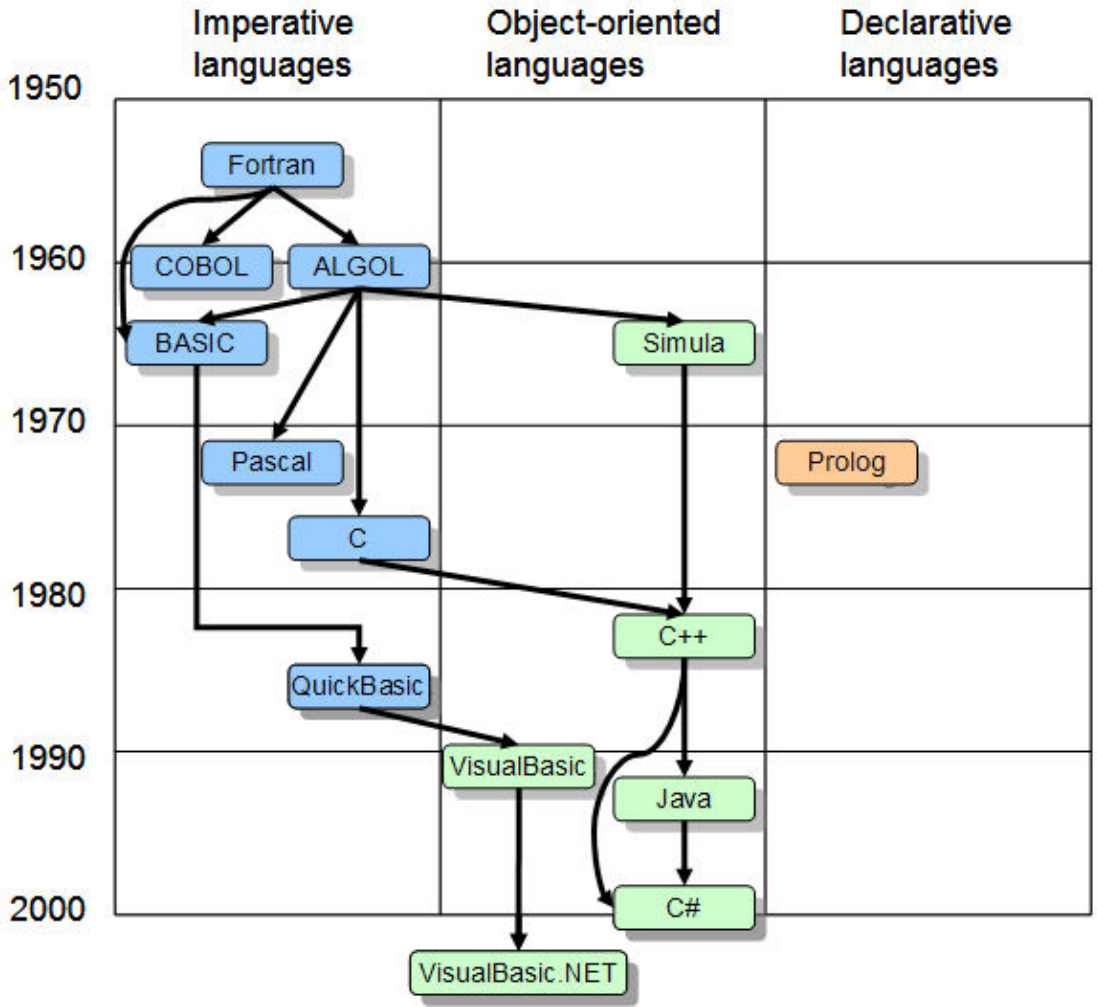
HKAL CS

B. Programming Languages		*(9 hours)*
• Evaluation of programming languages	• Students should develop a basic understanding of the programming paradigms involved in procedural, declarative, object-oriented, visual and query languages and be able to describe the criteria for selecting a programming language for problem-solving in a given context.	
• Program translation and stages of compilation	• Students should have a basic understanding of the need for high-level programming languages and the processes involved in the translation and analysis of such programs. Concepts include lexical, syntactic and semantic analysis, code generation, linkers and loaders, and compilers and interpreters.	

- Topic B is pitched to HKAL CS standard

Evolution of programming languages

- Low-level (Machine code, Assembly language) vs. High-level programming languages (3GL, 4GL, 5GL)



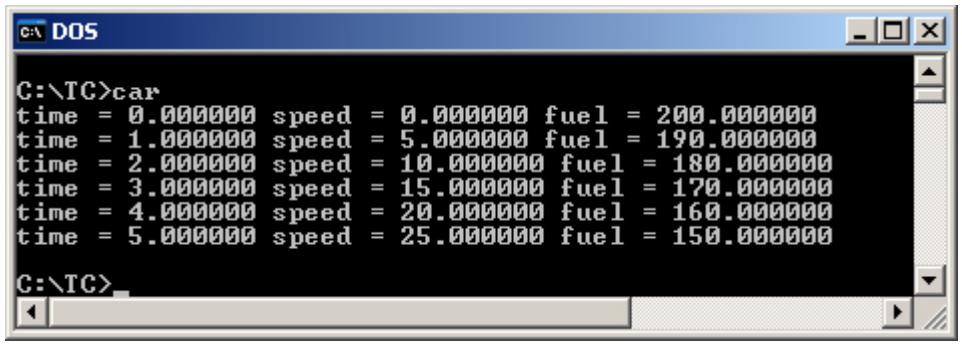
Programming paradigms

- Programming paradigms
  - Procedural (imperative)
  - Object-oriented
  - Declarative
  - Query (designed for DBMS; not generic and not a paradigm)

b.	**Programming Languages** (12 hours)		
i.	**Programming paradigms**	• Be aware of the evolution of programming languages. • Recognise the programming paradigms involved in procedural, logic, object-oriented and query languages. • Describe the criteria for selecting a programming language for a specific problem.	One programming language for each paradigm is selected for illustration.

Procedural programming paradigm

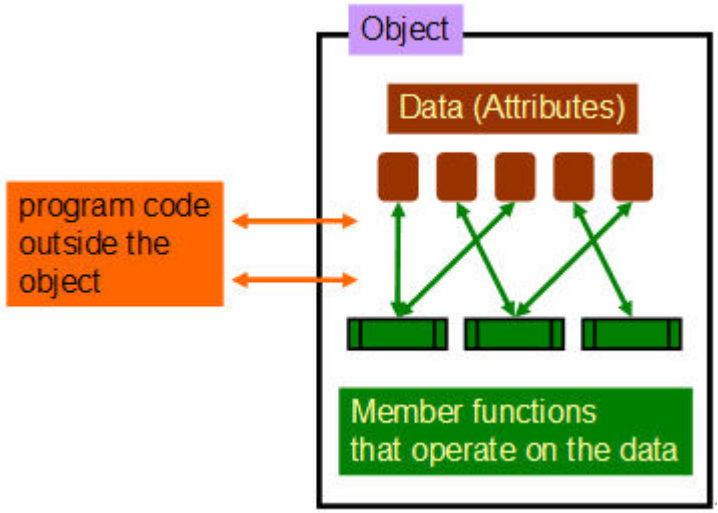
- Key concepts
  - Variables
  - Commands
  - Procedures

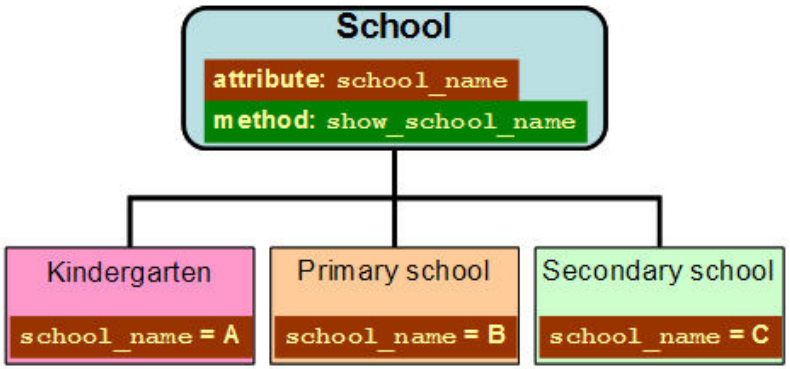


- Procedure is the representation of the method of solving a real-world problem
- Advantages and disadvantages

Object-oriented programming paradigm

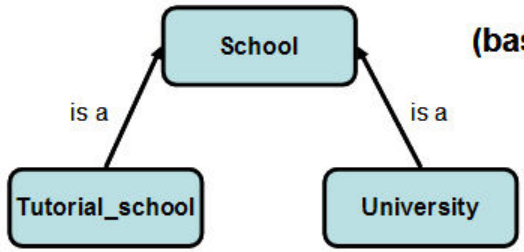
- Object and classes
- Encapsulation
- Inheritance
- Polymorphism





(Class)

(Objects)

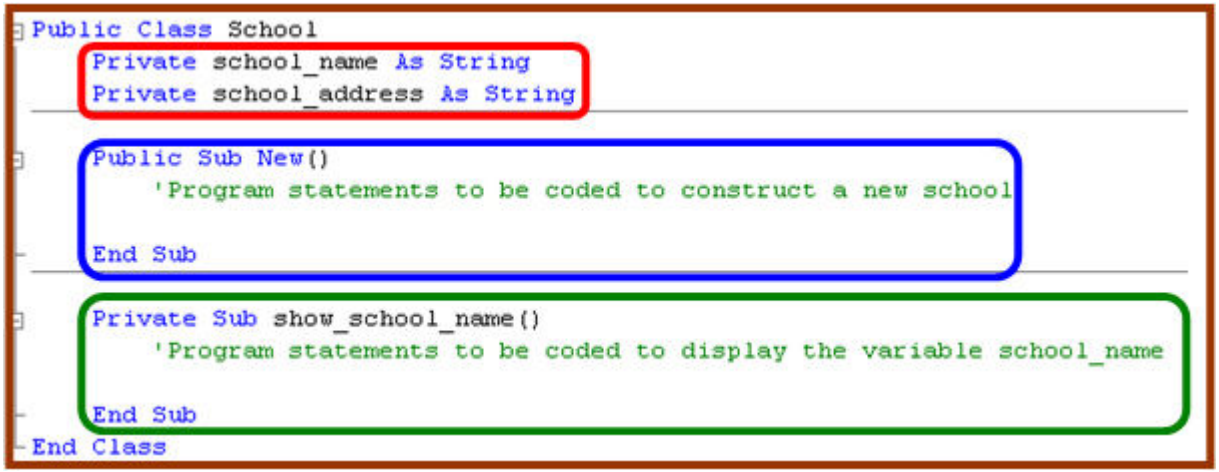


(base class/parent class)

(sub-class)

Object-oriented programming paradigm

- Object and classes
- Encapsulation
- Inheritance
- Polymorphism

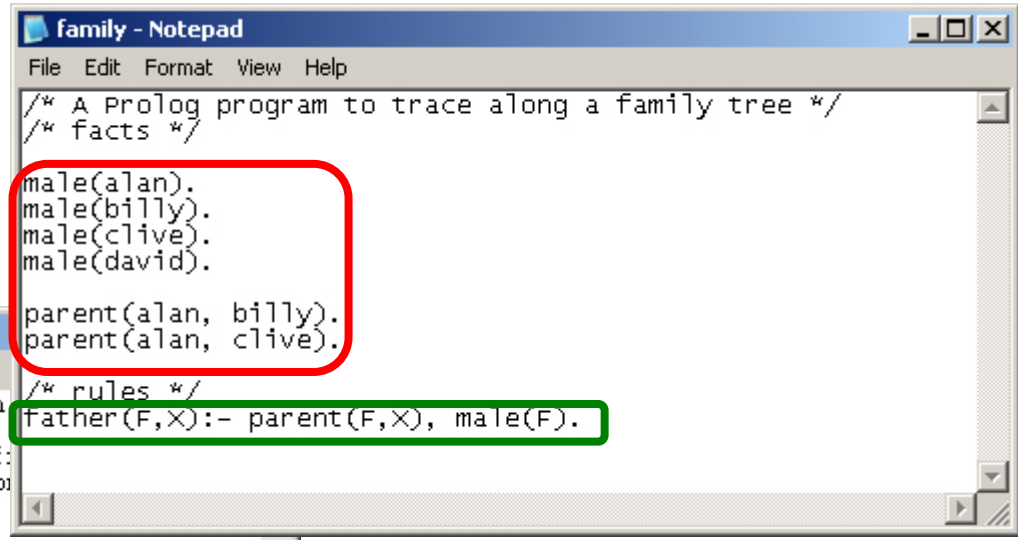


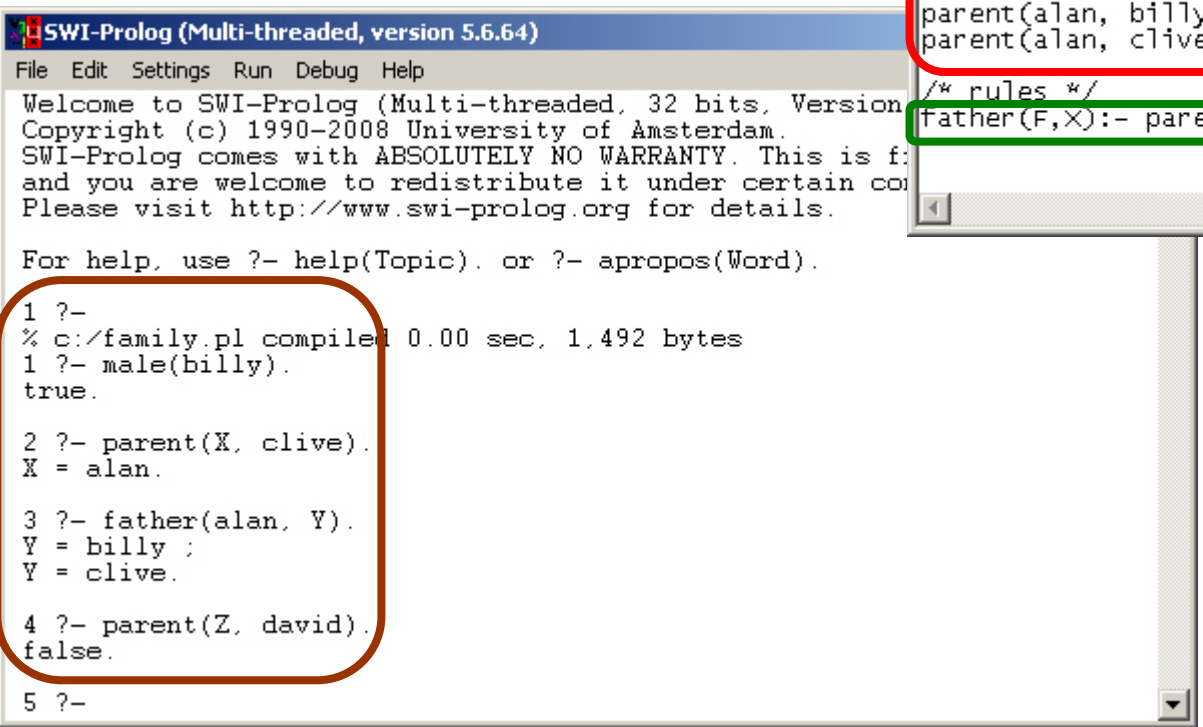
Declarative programming paradigm

- Students should practice so as to experience the difference between declarative programming paradigm and procedural/OO programming paradigms
- Recap 5GL is focused on “What to solve” rather than “How to solve”
- Free Prolog compilers available, e.g.
  - GNU Prolog
  - SWI Prolog

Prolog

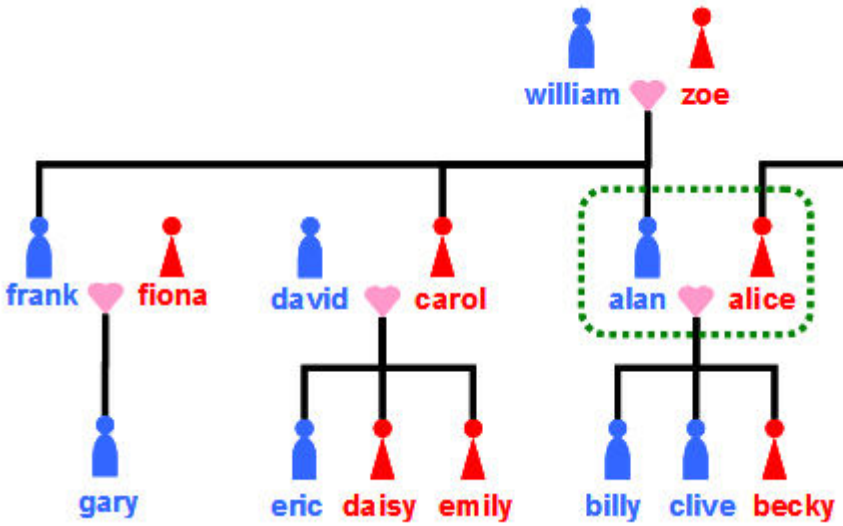
- Easy to start the hands-on practice
  - Notepad Editor / Built-in Editor
  - Enter “fact” and “rule”
  - Make Query





Prolog

- Learning Task B #1




```
SWI-Prolog (Multi-threaded, version 5.6.64)
File  Edit  Settings  Run  Debug  Help
1 ?- grandparent(X, gary).
X = william ;
X = zoe ;
false.

2 ?- grandparent(william, X).
X = billy ;
X = clive ;
X = becky ;
X = gary ;
X = eric ;
X = daisy ;
X = emily.

3 ?- cousin(X, becky).
X = gary ;
X = eric ;
X = daisy ;
X = emily ;
false.

4 ?-
```

- male, female
- father, mother, parent
- sibling
- grandparent, cousin

Query language

- Experience query language by practicing

Employee : Table

Employee_ID	Name	Sex	DOB	Department
07001	Wong Tai Man	M	1985.01.13	Administration
07002	Lee Siu Mei	F	1984.11.25	Accounting
07003	Lau Bo Bo	F	1983.07.31	Sales
08002	Wong Lok Tin	M	1987.04.09	Sales
08003	Lee Chi fai	M	1986.12.10	Accounting
09001	Chan Amy	F	1983.09.17	Sales

Record: 1 of 6

```
SELECT Name, Department
FROM Employee
WHERE Sex ='F'
ORDER BY Name;
```

Employee Query1 : Select Query

Name	Department
Chan Amy	Sales
Lau Bo Bo	Sales
Lee Siu Mei	Accounting

Record: 1 of 3

Criteria of selecting a programming language

- No universal rules, but some criteria are provided for consideration
- Justify decision after comparing several candidate programming languages
- Case study: Learning Task B #2, Assessment Task B #1

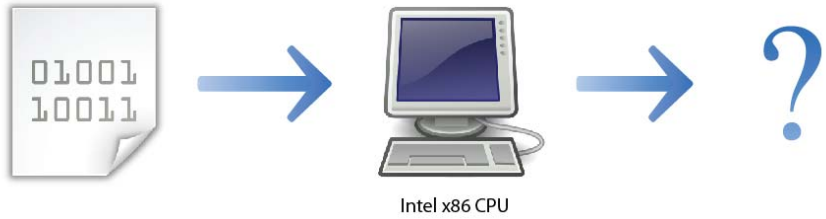
Criteria of selecting a programming language

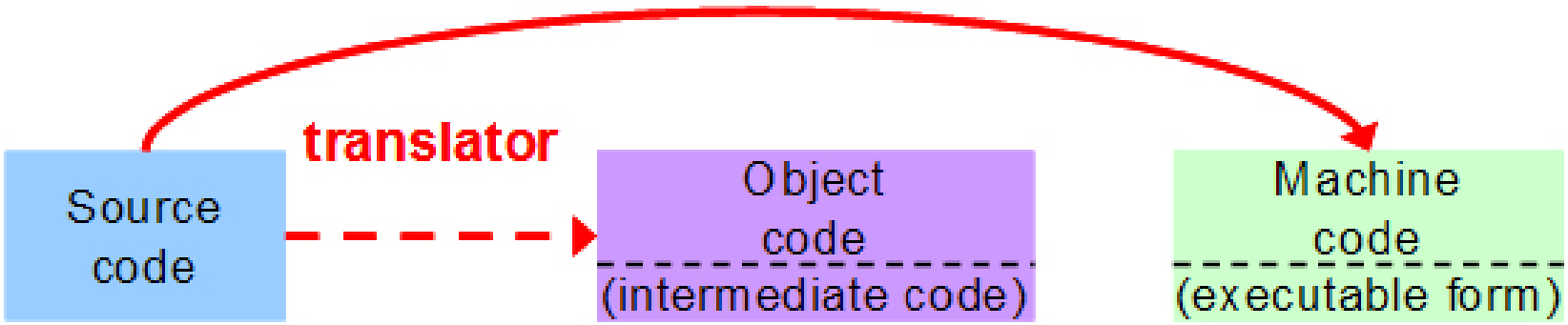
- Learning task: Develop a small-scale library system
- Assessment task: Develop a program to be used in sports day

Criterion	Description	Relevant requirement code	Findings	Pascal	C	Visual Basic	Java
Execution efficiency	It refers to the efficient execution of non-I/O computational operations.	B2	C language compiler can generate efficient executable code.		✓		
Portability	It is used in the narrow sense that the source code can be easily compiled to executable codes on different operating systems.	C1	C language and Java have very good portability across a wide range of computer platforms.		✓		✓
Readability	It means whether a program written in a language can be read and understood with ease.	C2	As long as a programmer has a good programming habit, the code should be easily understood by others, no matter which programming language he uses to code.	✓	✓	✓	✓
Utility libraries and development tools	It concerns the availability of a wide range of utility libraries and efficient development tools for a computer language. This can affect the application development time significantly.	G1	C, VB and Java have plenty of utility libraries to use.		✓	✓	✓

Topic B2: Language translators and compilers

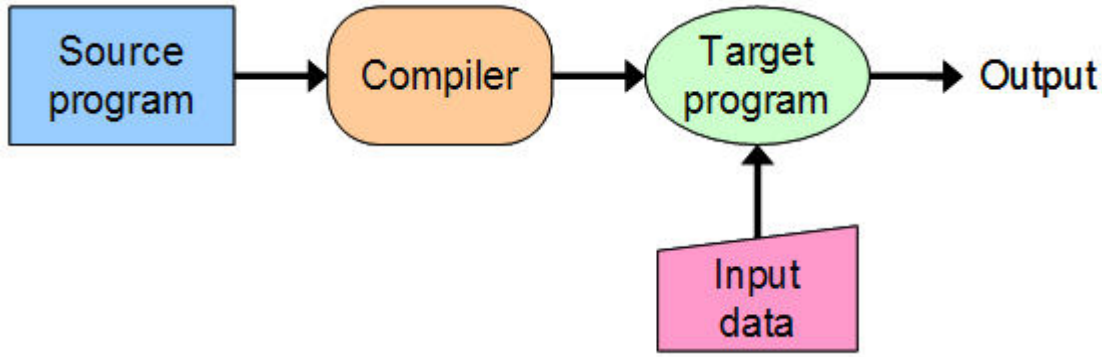
- Why translator is needed?
- Source code, intermediate code, executable code

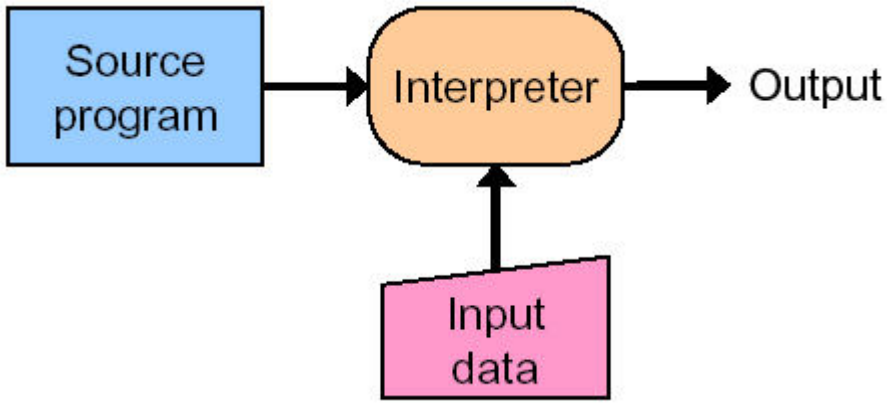




Compilers and Interpreters

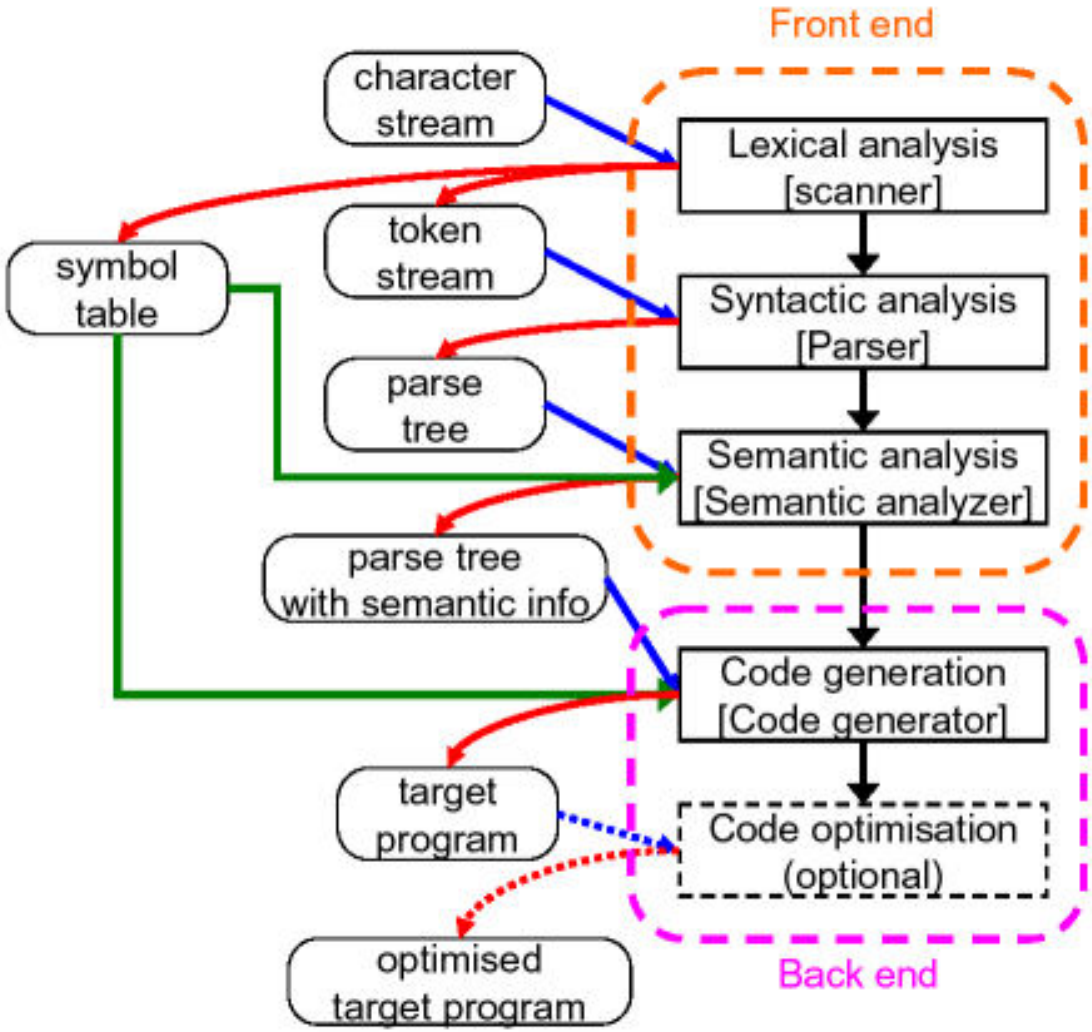
- What a compiler does and what an interpreter does?
- Compare compilers and interpreters





Overview of phases of compilation

- HKDSE sample paper 2D Q4



Lexical analysis

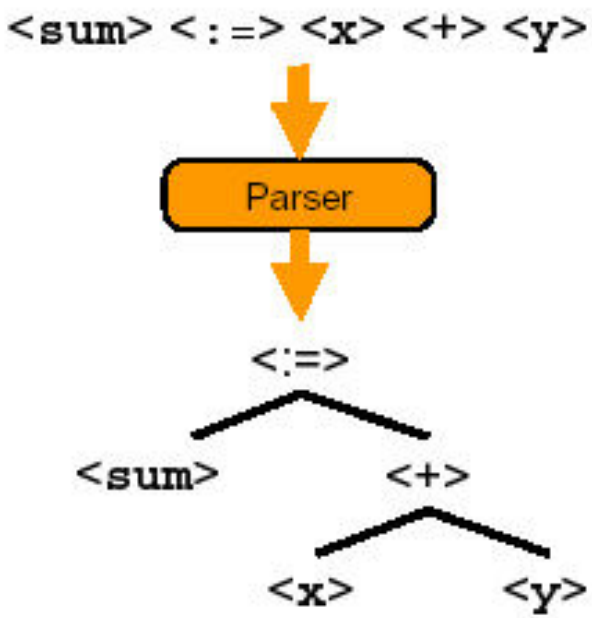
- What a lexical analyzer does?
- Character stream from source code ➔ Tokens + Symbols table

```
program add
[Calculate the sum of 2 integers]
var sum, x, y : integer
start
   input(x, y)
   sum := x + y
   output(sum)
end
```

program	add	var	sum	,	x	,	y
:	integer	start	input	(	x	,	y
sum	:=	x	+	y	output	(	sum
)	end						

Parser

- What is syntactic analysis?
- Tokens ➔ Parse tree

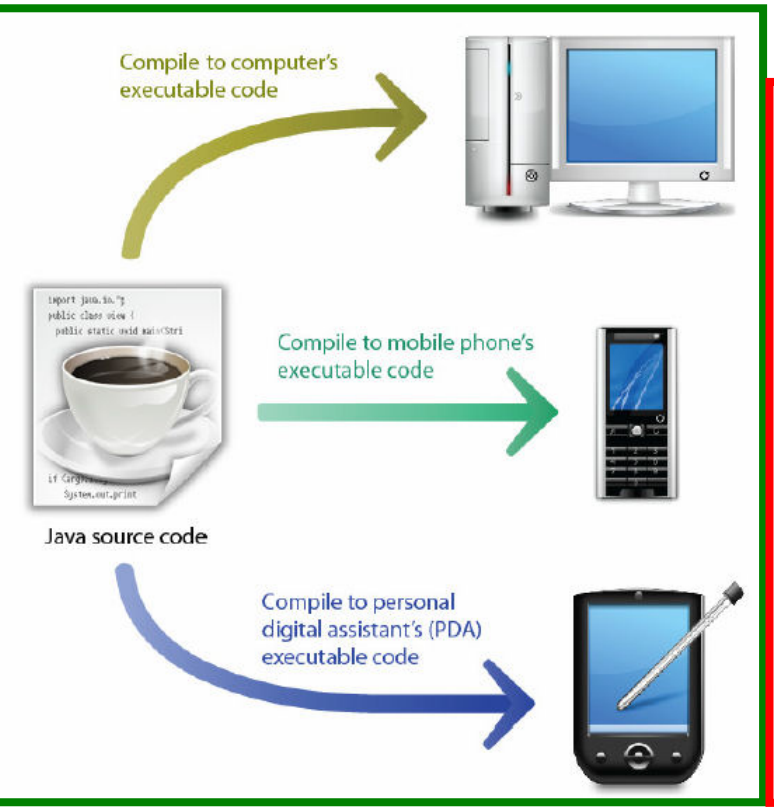


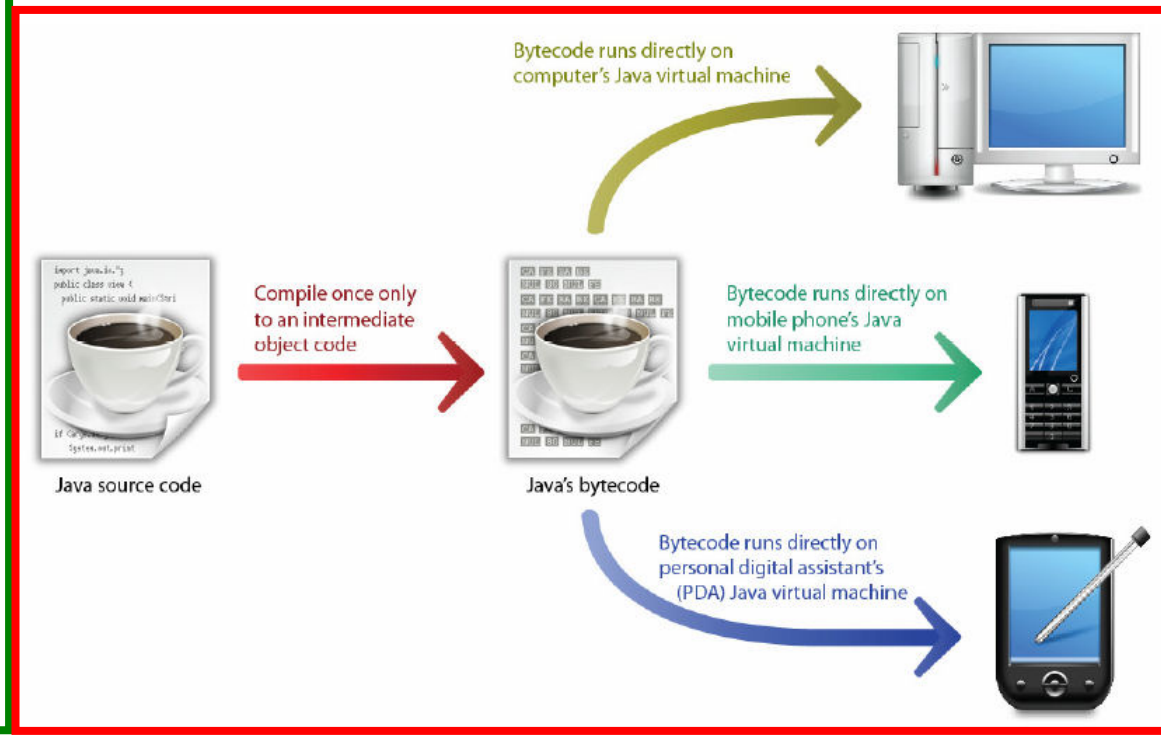
Semantic analyzer

- What a semantic analyzer does?
- Parse tree + Symbols ➔ Parse tree with semantic information
- Front end

Back end of a compiler

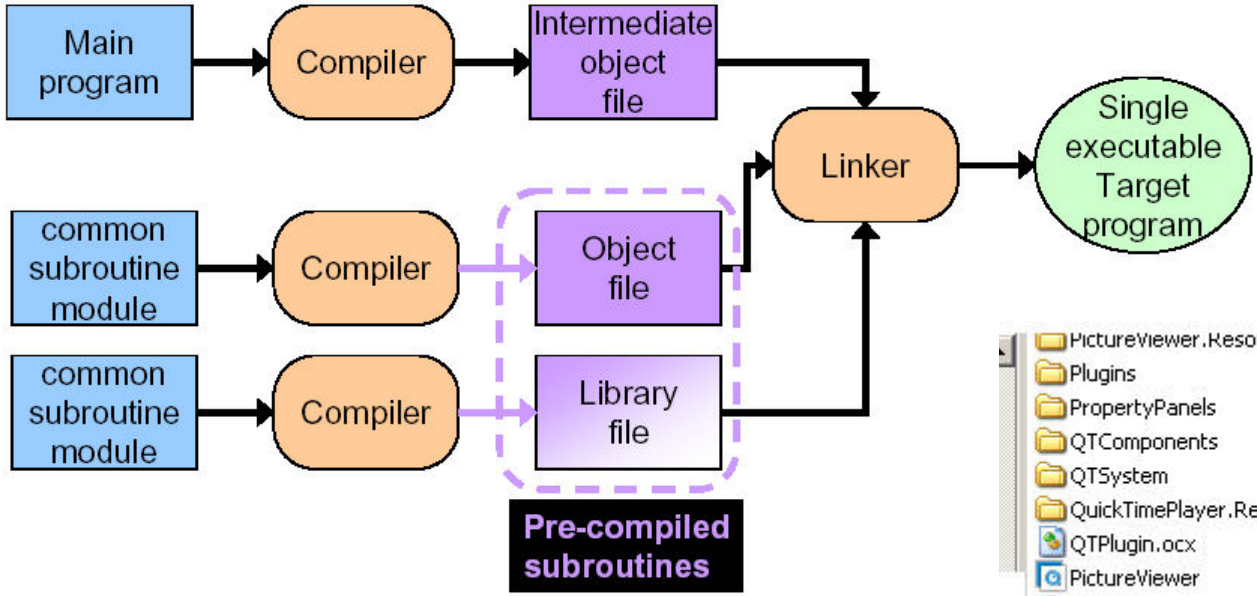
- Code generation + Code optimization
- Intermediate code (e.g. Java bytecode)
- Java Virtual Machine (JVM)

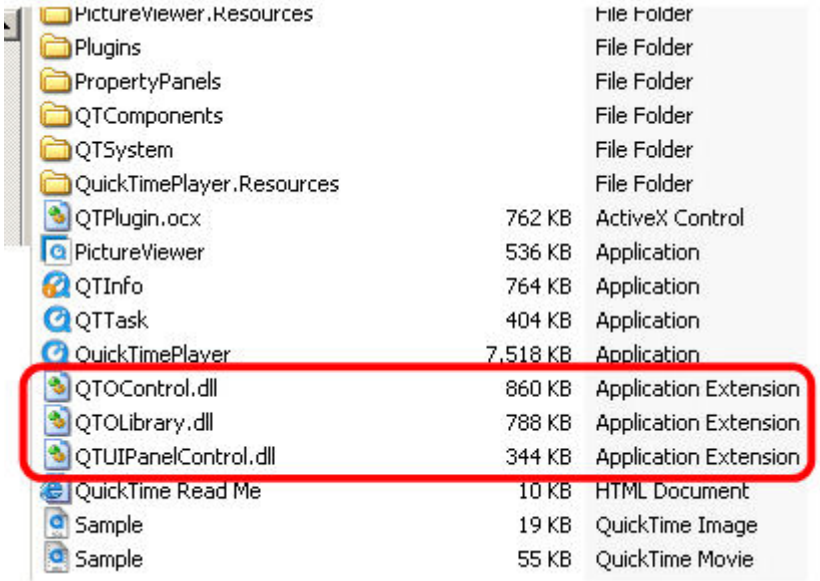




Linker and loader

- Functions of a linker and a loader
- Dynamic linking (DLL)





Assessment Task B #2

Chapter B2 Language translators and compilers

Exercise / Quiz

1. (a) The source code of a program written in a high level language must be translated to an object code before the program can be executed on a computer. What is meant by "source code" and "object code"?

(b) What are the purposes of a compiler?

(c) Suggest any TWO reasons why almost all programs are usually compiled before they are sold to customers.

The End
Thank you

Pass to “Highlight on Java”

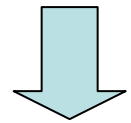
PROMOTION SESSION ON NSS ICT ELECTIVE –

Highlights on Java

Centre for Information Technology in Education, HKU

Elective Option D: Software Development

- Topic A on “Programming” (47 hours)
  - Choose a programming language from Pascal, C, Visual Basic, Java
- Topic B on “Programming Languages” (12 hours)
  - Programming paradigms
  - Language translators and compilers



Topic A: **Java** Programming

1	Getting Familiar with the Programming Environment	3 periods	2 hours
2	Input and Output	3 periods	2 hours
3	Arithmetic Operations and Functions	4½ periods	3 hours
4	Selection Structure	6 periods	4 hours
5	Iteration Structure	7½ periods	5 hours
6	**Programming Modules in Java**	9 periods	6 hours
7	Arrays	6 periods	4 hours
8	Strings	4½ periods	3 hours
9	File Handling	6 periods	4 hours
10	Searching, Sorting and Merging	10½ periods	7 hours
11	Stacks, Queues and Linked Lists	10½ periods	7 hours
	Total:	70½ periods	47 hours

Java is an Object Oriented Language Yet still allows programming With Modules/Methods !!

Resource package

- 11 Chapters (A1 – A11)
- 4 Learning tasks
  - A1 – A3: Learning Task A #1
  - A4 – A6: Learning Task A #2
  - A7 – A9: Learning Task A #3
  - A10 – A11: Learning Task A #4
- 4 Assessment tasks
  - A1 – A3: Assessment Task A #1
  - A4 – A6: Assessment Task A #2
  - A7 – A9: Assessment Task A #3
  - A10 – A11: Assessment Task A #4
- Teaching plans for each chapter
- Source programs of examples and practical tasks

Highlight on Java

- Programming Languages of Choices : Pascal, C, Visual Basic, Java
- Numerous resources for Pascal and C
- Highlights on **Visual Basic** and **Java**

Highlight on Java IDE - NetBeans

- NetBeans Version 6.7.1 or Later
  - http://www.netbeans.org

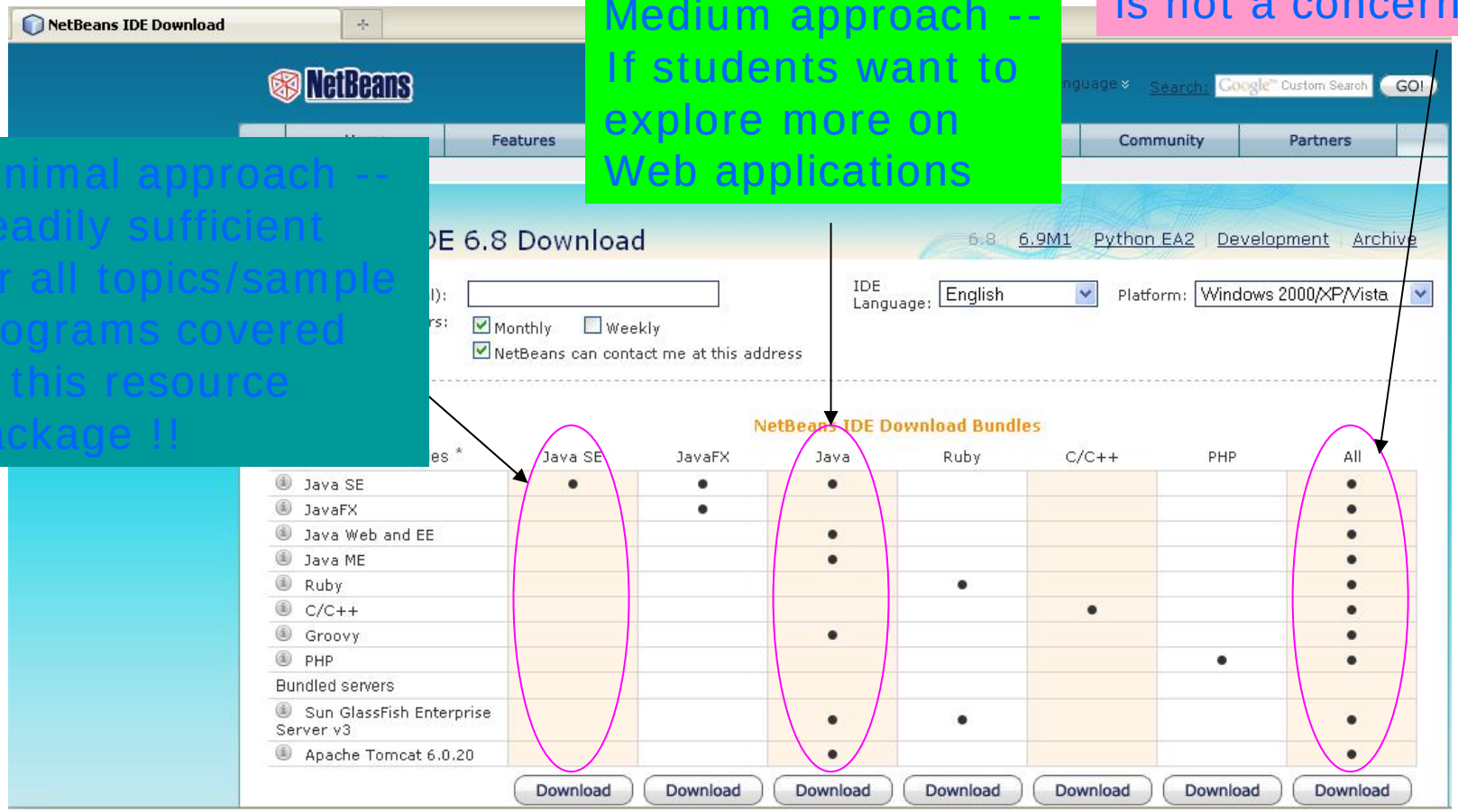
Highlight on Java IDE - NetBeans

- Notes on Downloads

Maximal approach -- Only if disk space is not a concern !!

Medium approach -- If students want to explore more on Web applications

Minimal approach -- Readily sufficient for all topics/sample Programs covered in this resource Package !!



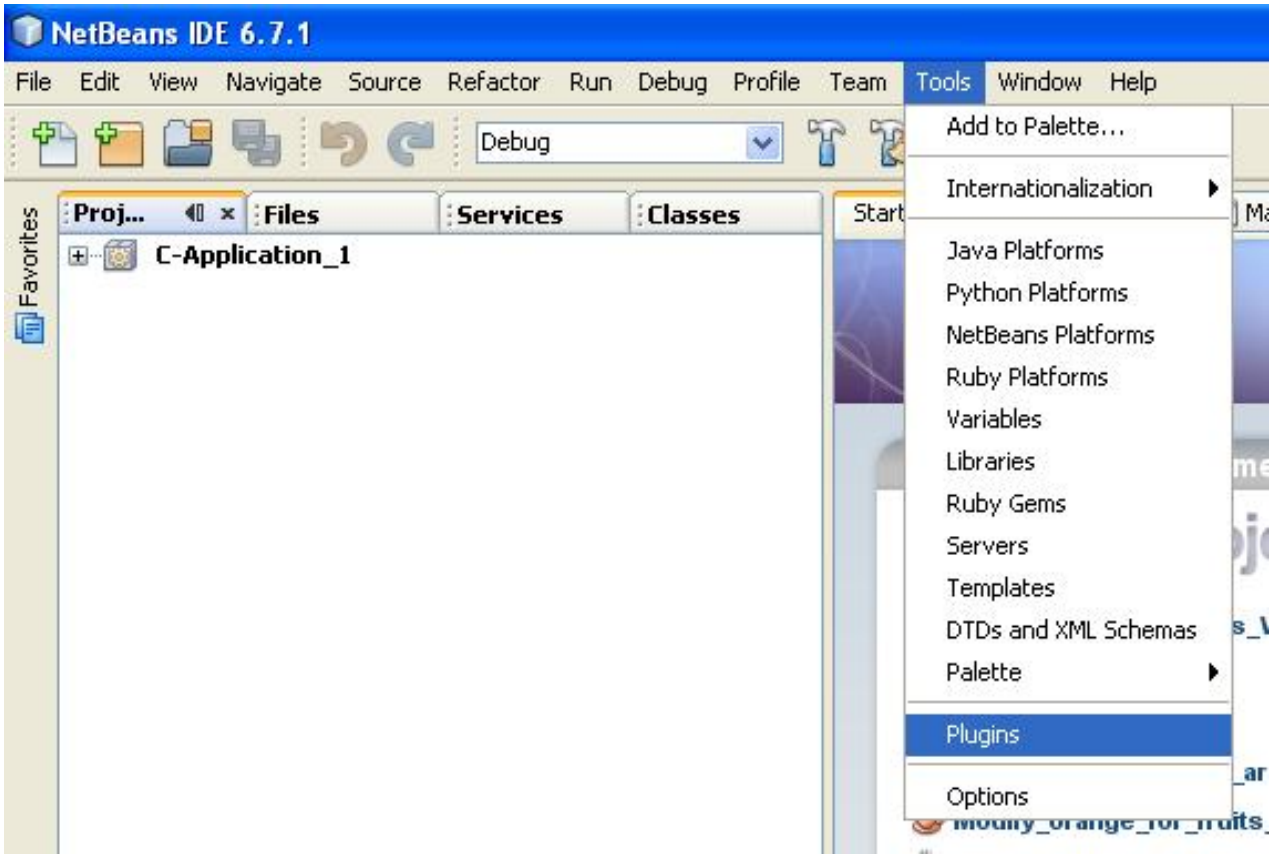
Highlight on Java IDE - NetBeans

- If your download starts with the ‘minimal’ approach, one can always update with the additional features (such as C++ or PHP) as ‘Plug-ins’ later.

- In fact, all available features in Netbeans are loaded incrementally as “modules” during program

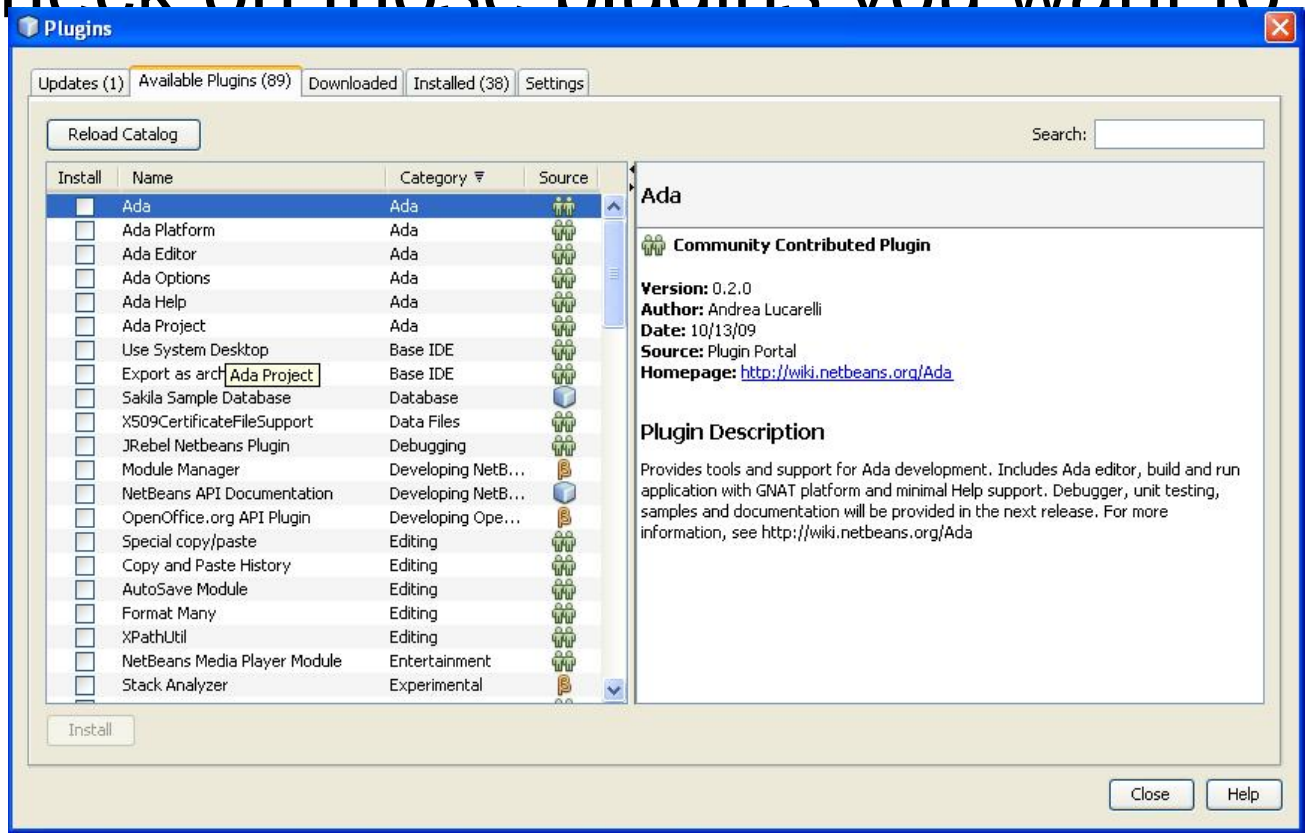
Highlight on Java IDE - NetBeans

- Updates on Plug-ins:: go to the “Tools” menu -> click on “Plugins”



Highlight on Java IDE - NetBeans

- Next, the IDE “Plugins” window will pop up. Simply click on the “Available Plugins” and check on those plugins you want to install.



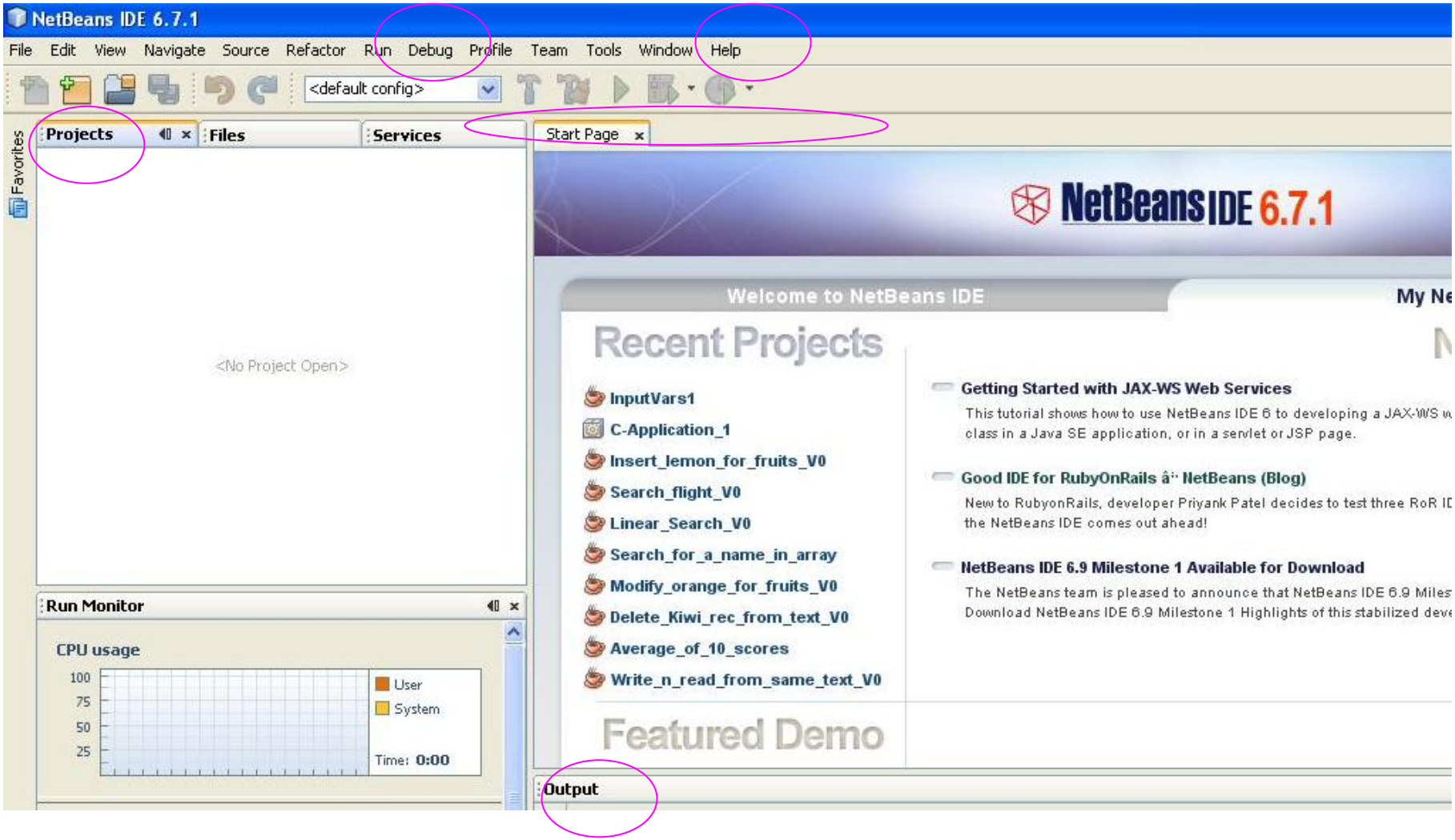
Highlight on Java IDE - NetBeans

- After the required plugins are downloaded and installed, it will prompt for restarting of the IDE again.

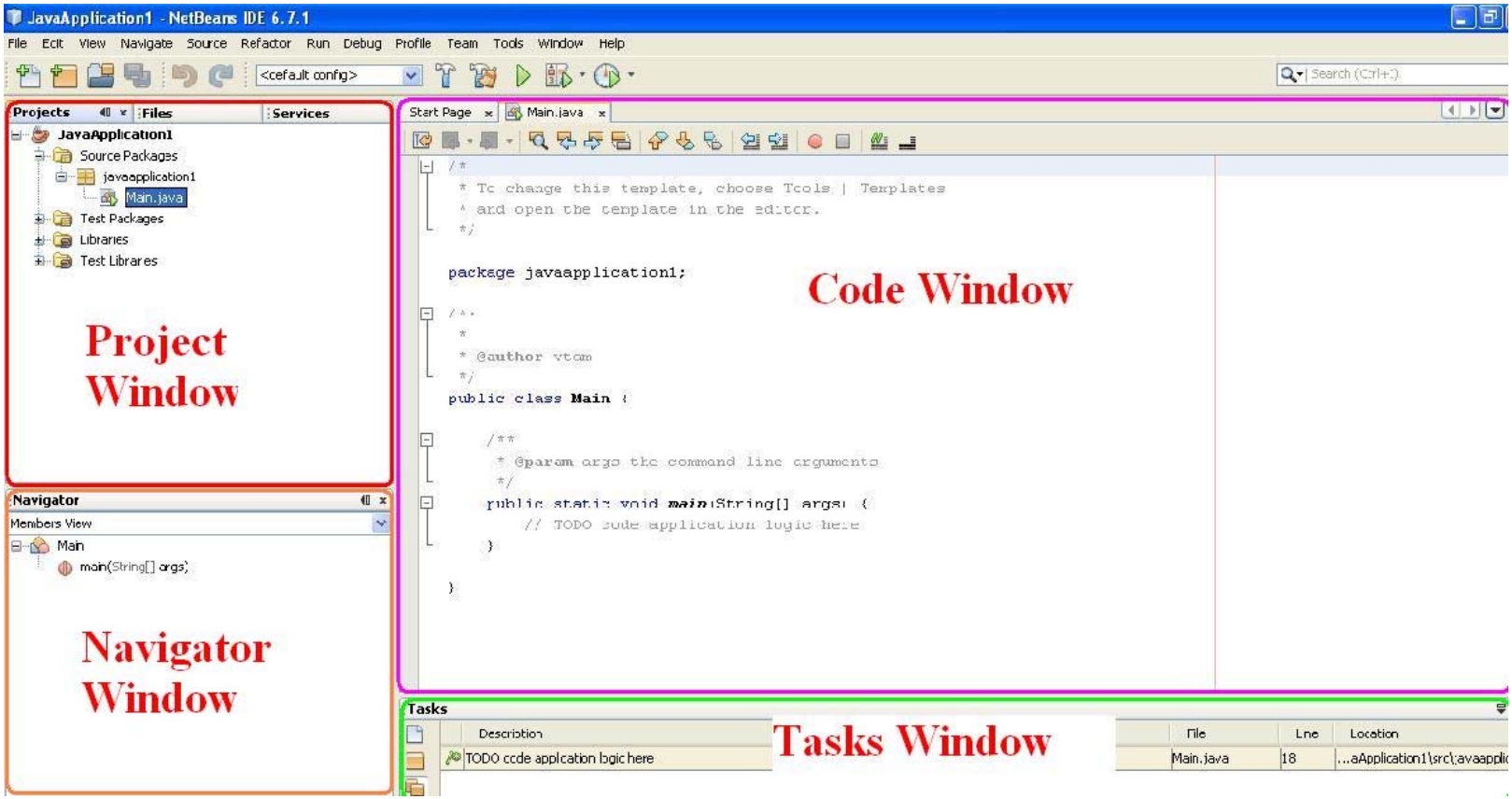
Minimalist approach

- Students start as soon as possible on some meaningful tasks
  - Understand basic instructions with a flat learning curve
  - Extensive Uses of NetBeans IDE facilities including the Project Window, Code Window, Output Window, “Debug” function and auto-filling function for suggestions of concerned Java packages, methods and formats of input parameters, and so on.

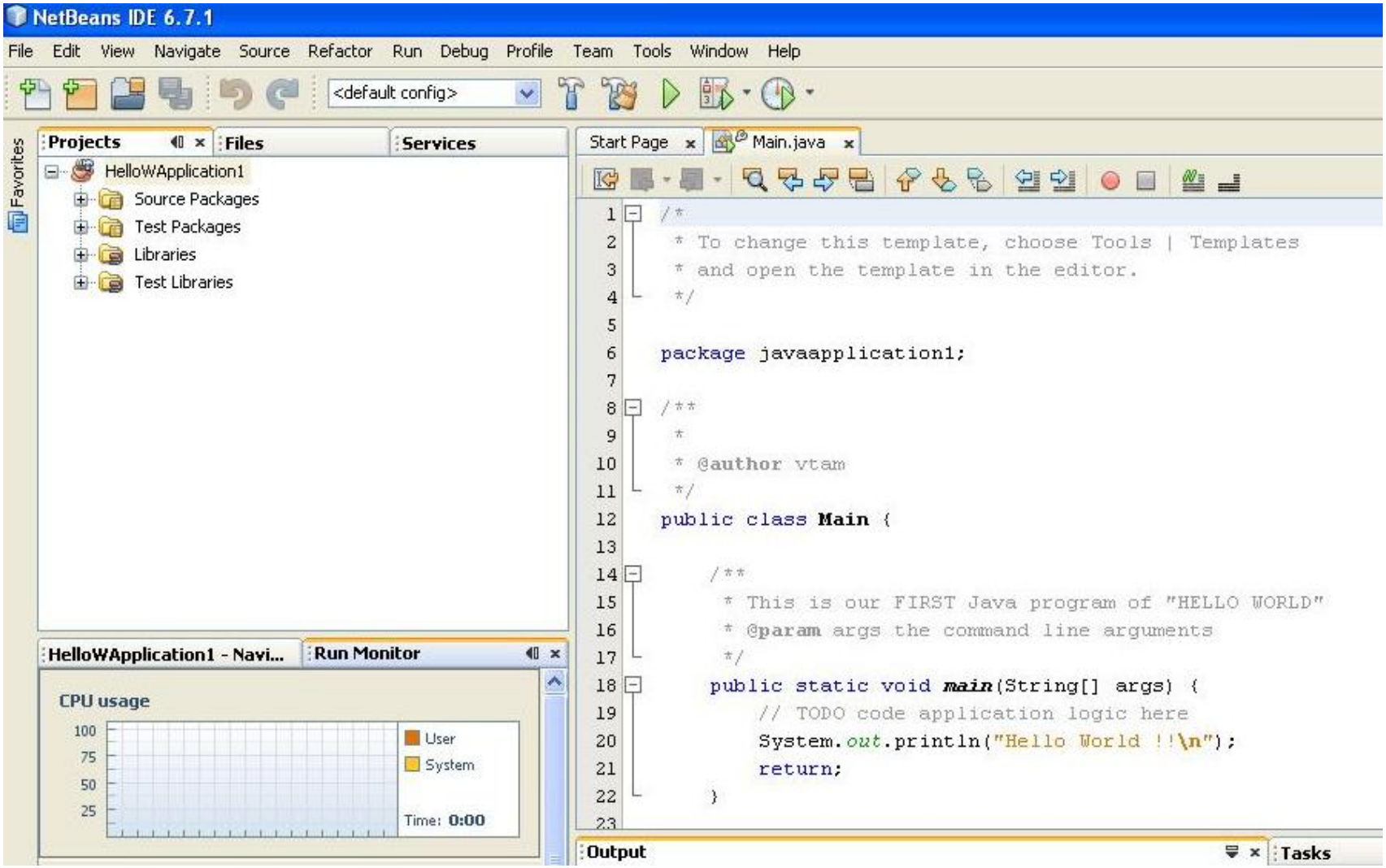
The starting IDE interface..



More on the IDE interface…



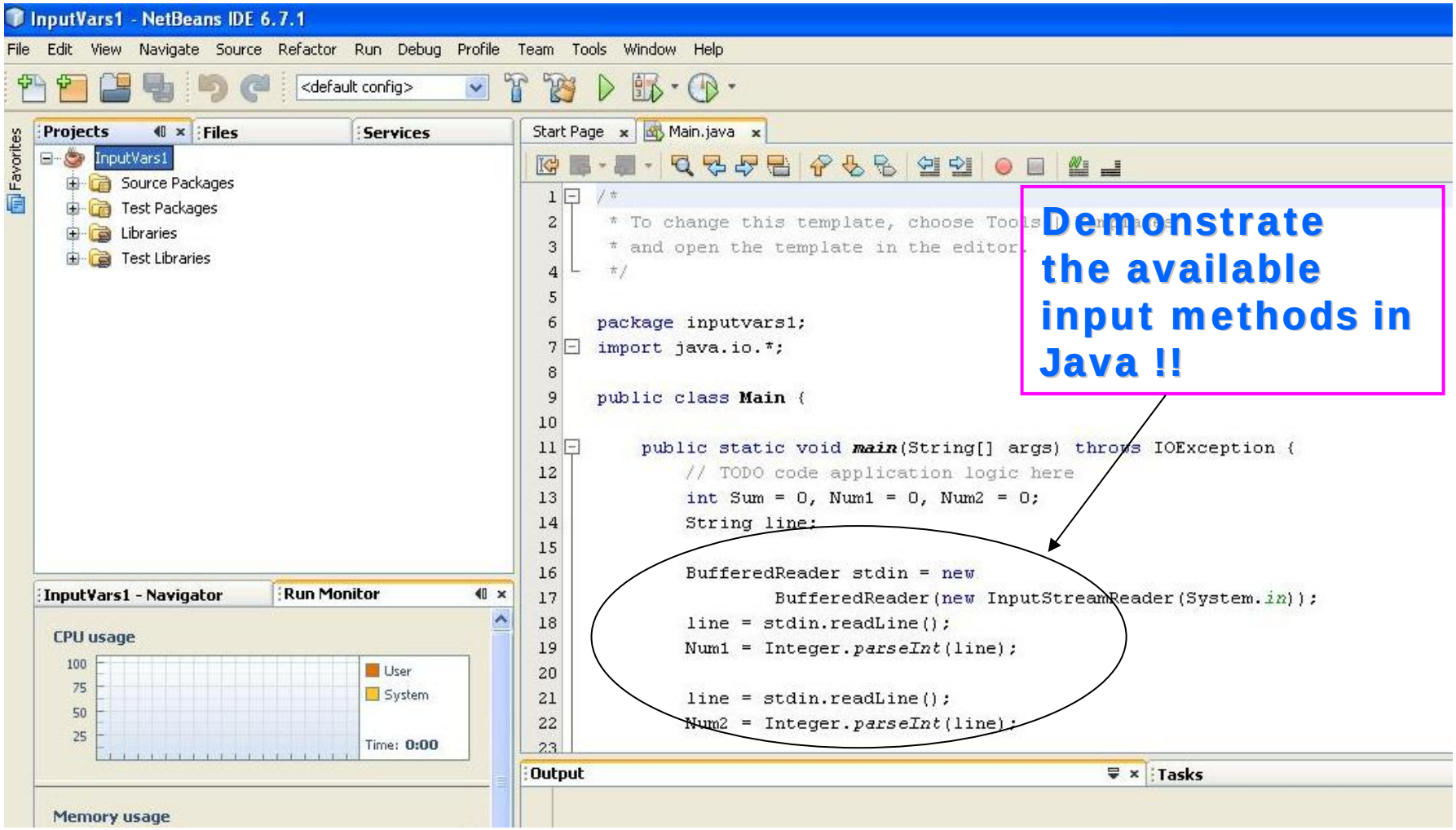
Starting with Simple Java programs for ease of learning and “confidence”..



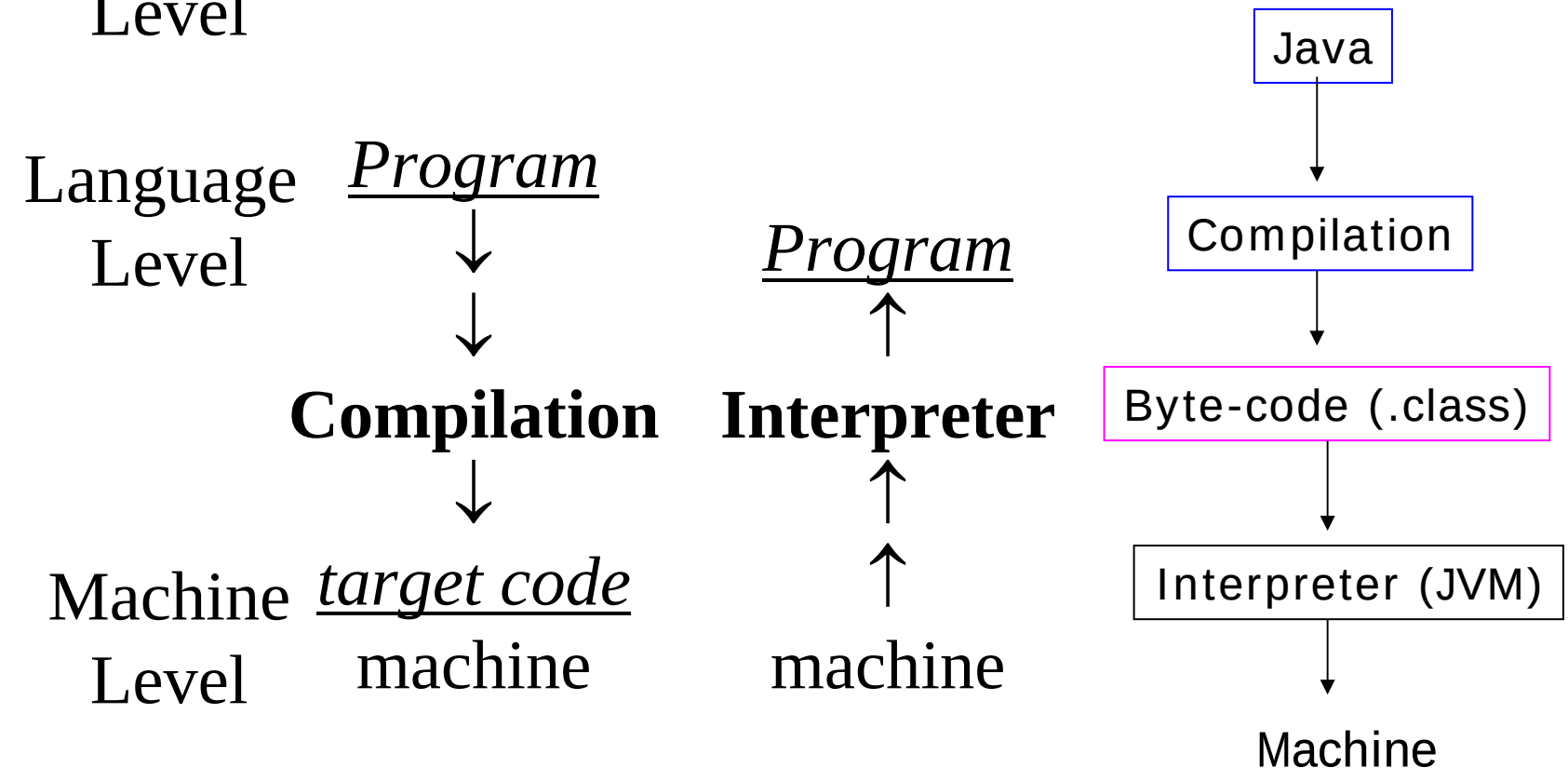
Starting with Simple Java programs for ease of learning and “confidence”..

- As in most programming paradigms or reference, “Hello World” is often the first and simplest sample program that is sufficient to DEMONSTRATE::
  - The minimal library files/packages that must be included to write a (Java) program;
  - The basic function/method to display/output the “Hello World” message (a constant Java String);
  - How the program interact/communicate with its underlying O.S. platform, i.e. through the main() method of the user-defined class !!

Starting with Simple Java programs for ease of learning..



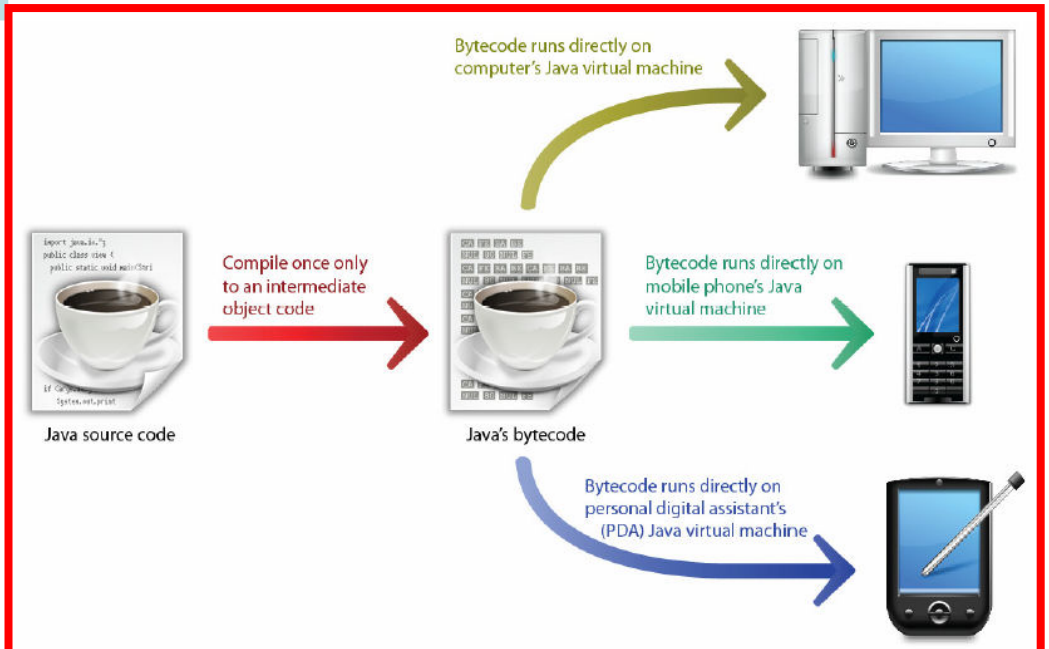
Different Abstraction Levels of Programming..



Java Compiler + Interpreter (JVM)

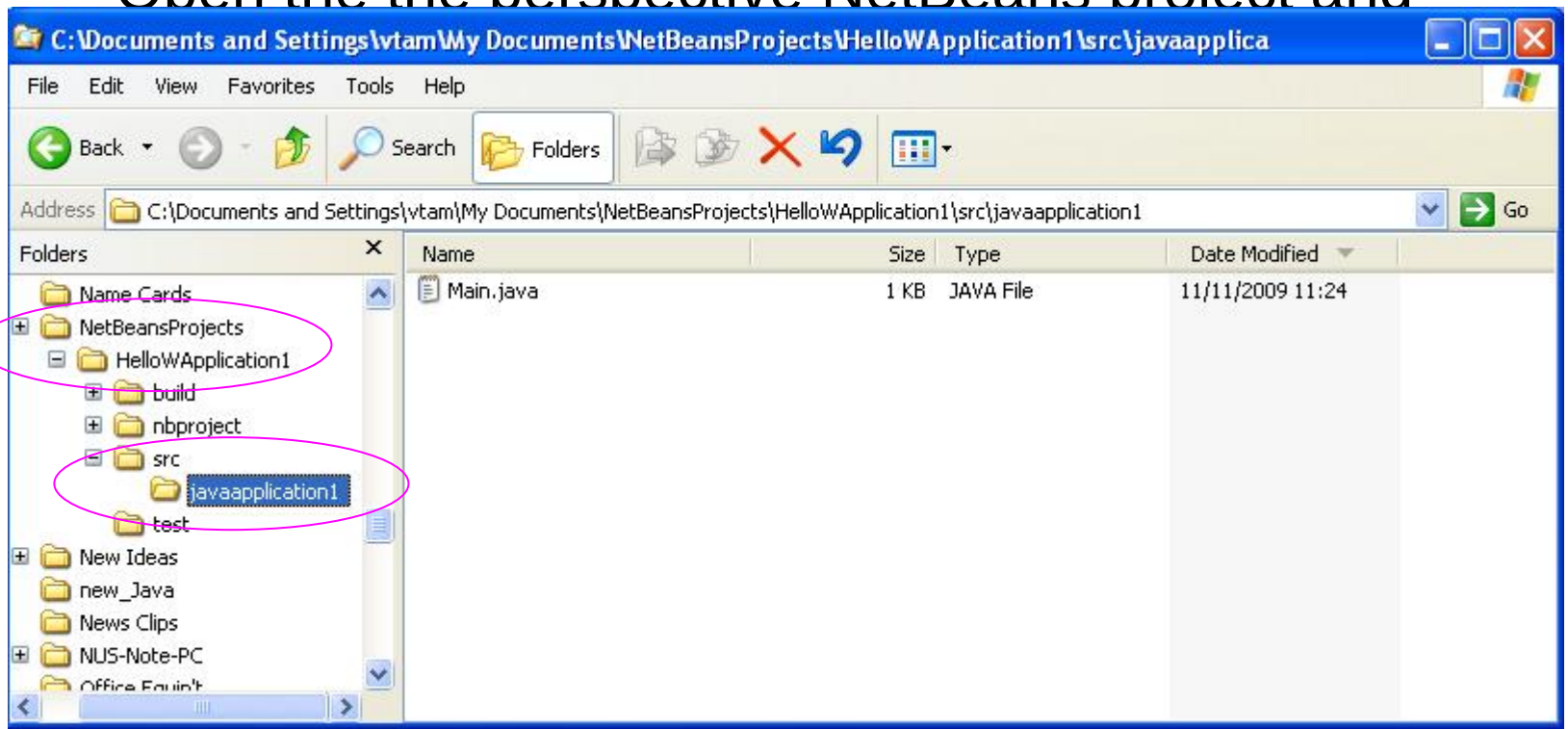
= Portability (or Platform Independence) !!

- Code generation + Code optimization done by the Java compiler [javac.exe in the JDK]
- Intermediate code (e.g. Java bytecode) to be interpreted and executed by the Java Virtual Machine [java.exe]



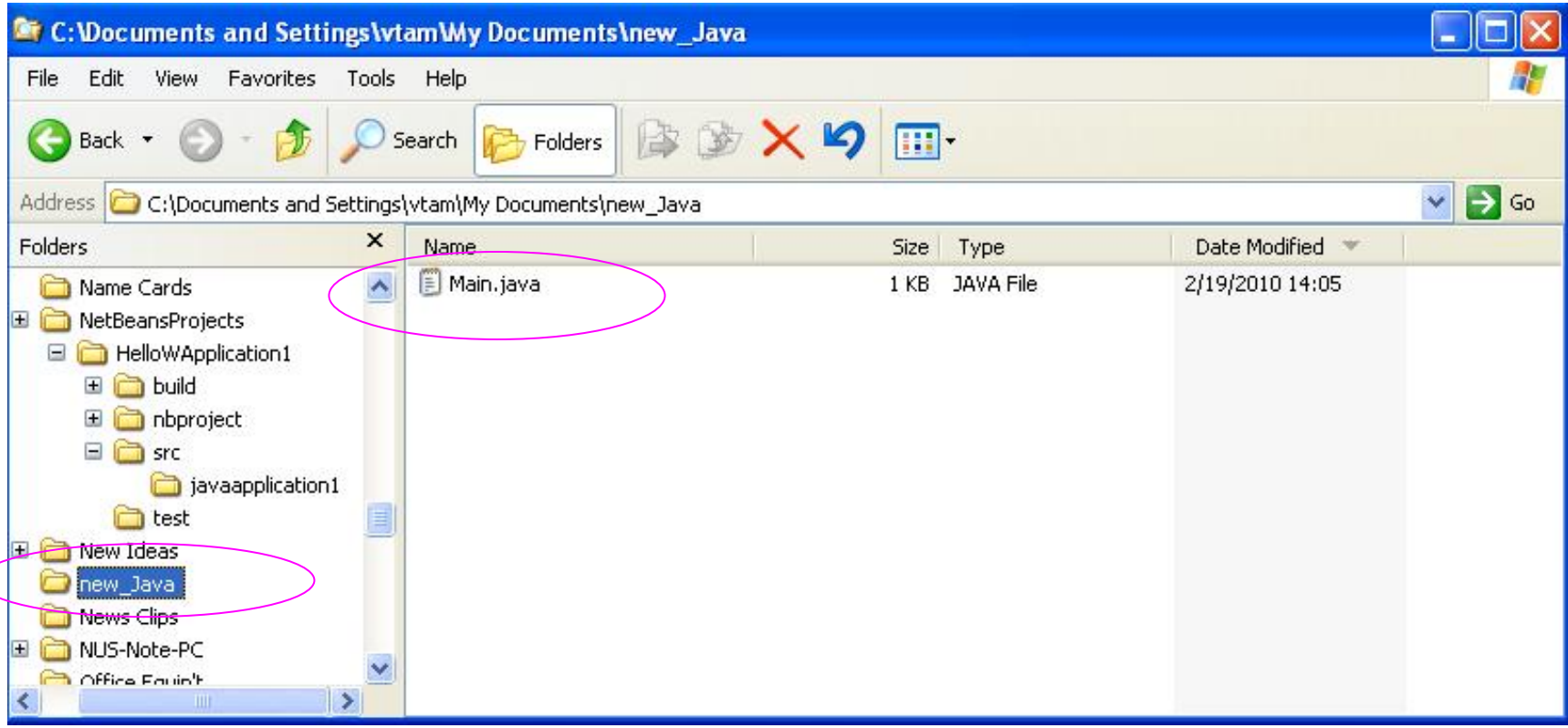
Demo. of Compilation + Interpretation…(Using Commands)

- Teachers may consider to perform a demo. to the students to clearly illustrate the following steps:
  - Open the the perspective NetBeans project and



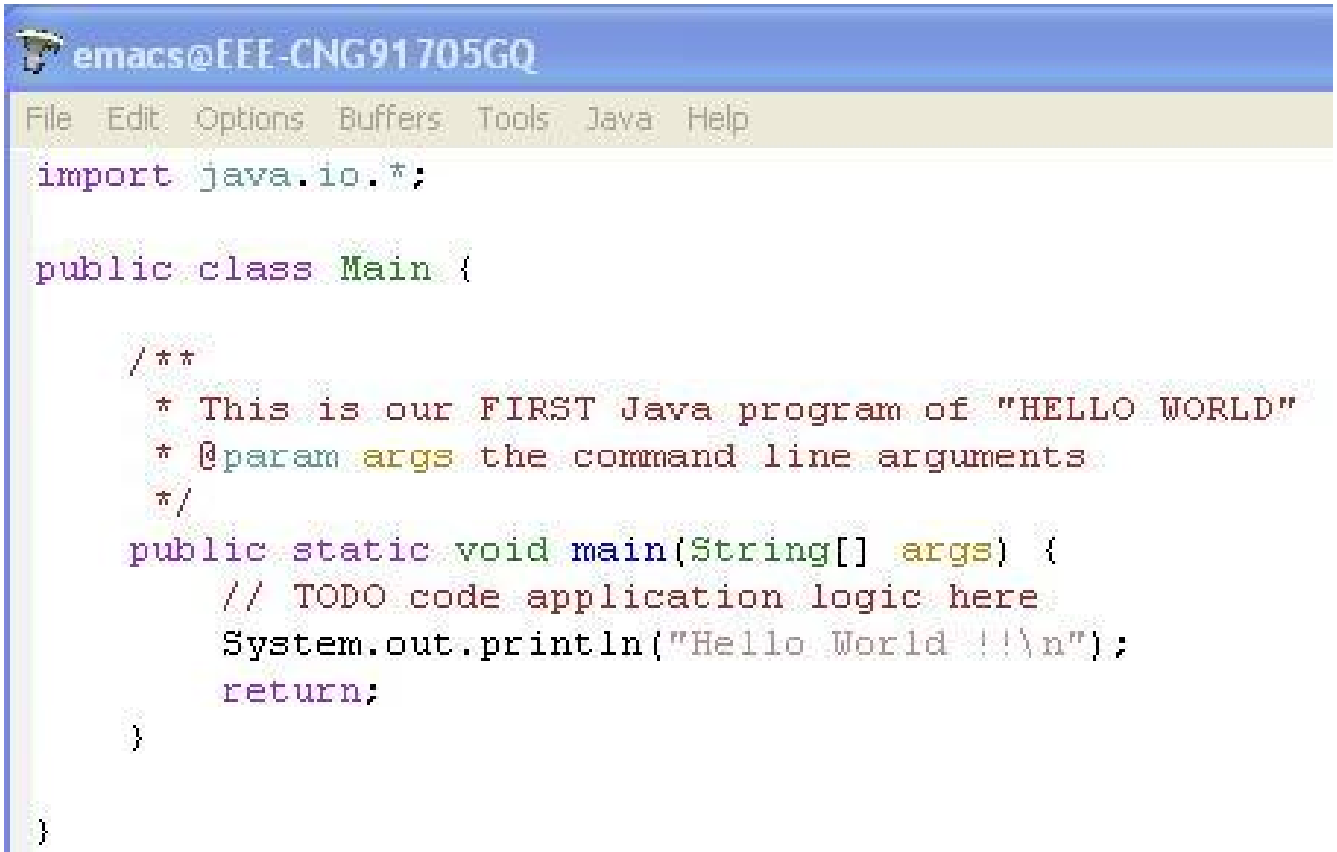
Demo. of Compilation + Interpretation…(Using Commands)

– Copy the “Main.java” to another new directory;



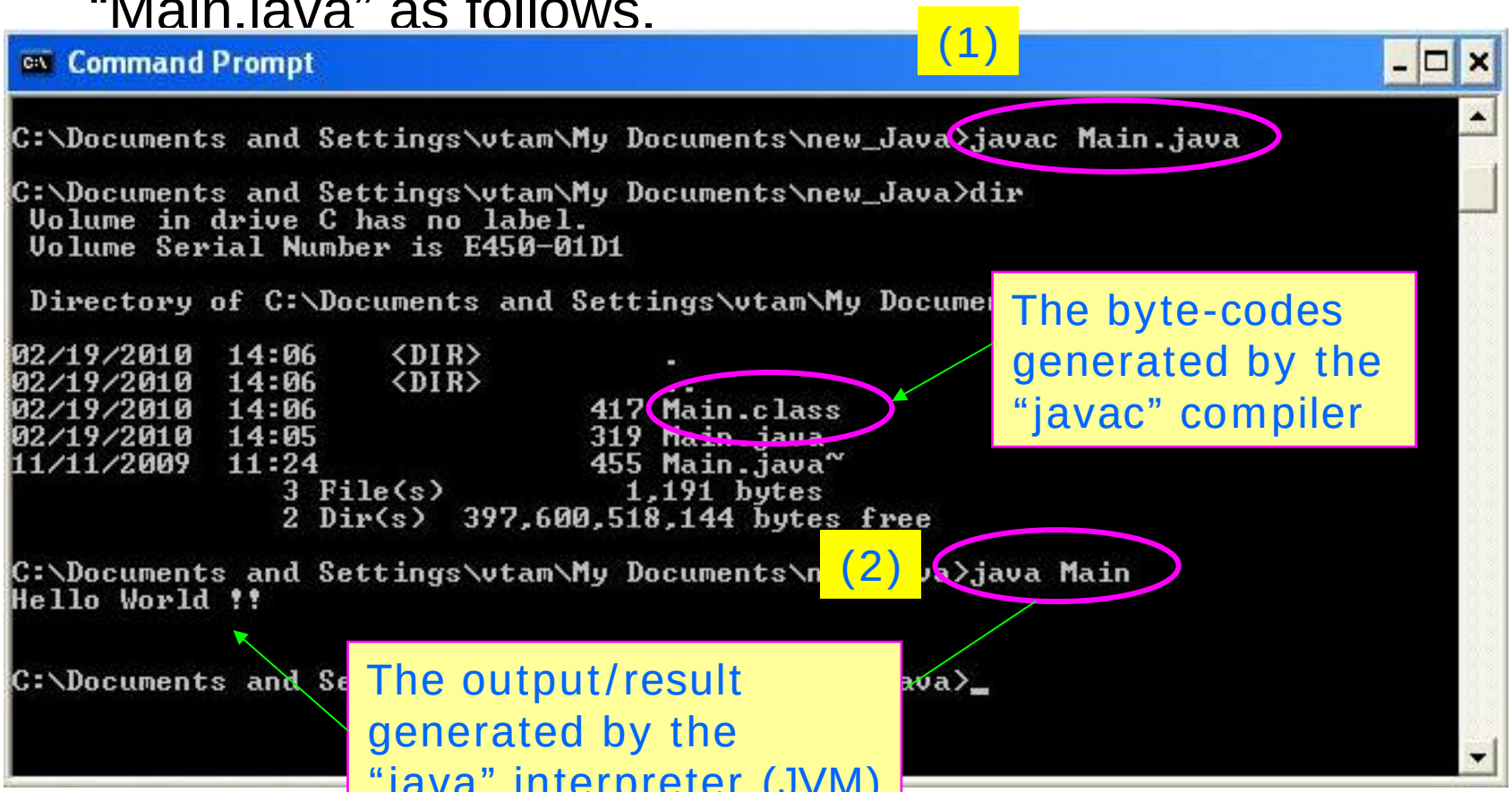
Demo. of Compilation + Interpretation…(Using Commands)

– Edit the “Main.java” (using Notepad or your favorite text editor) in the new directory as below, and then SAVE the new version:



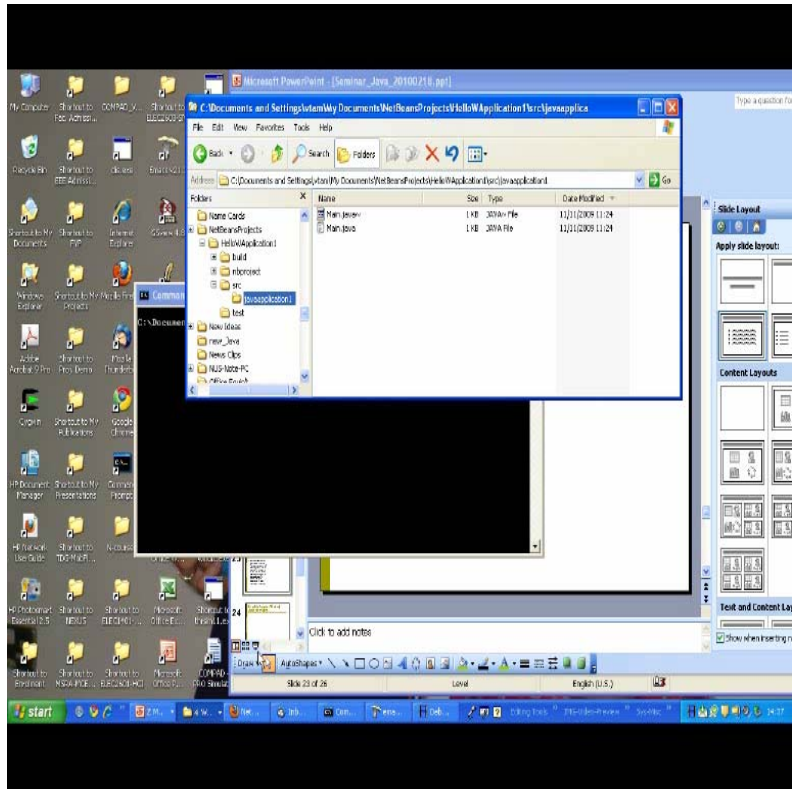
Demo. of Compilation + Interpretation…(Using Commands)

– Now, open a command prompt window and issue commands to compile & execute the saved "Main.java" as follows.



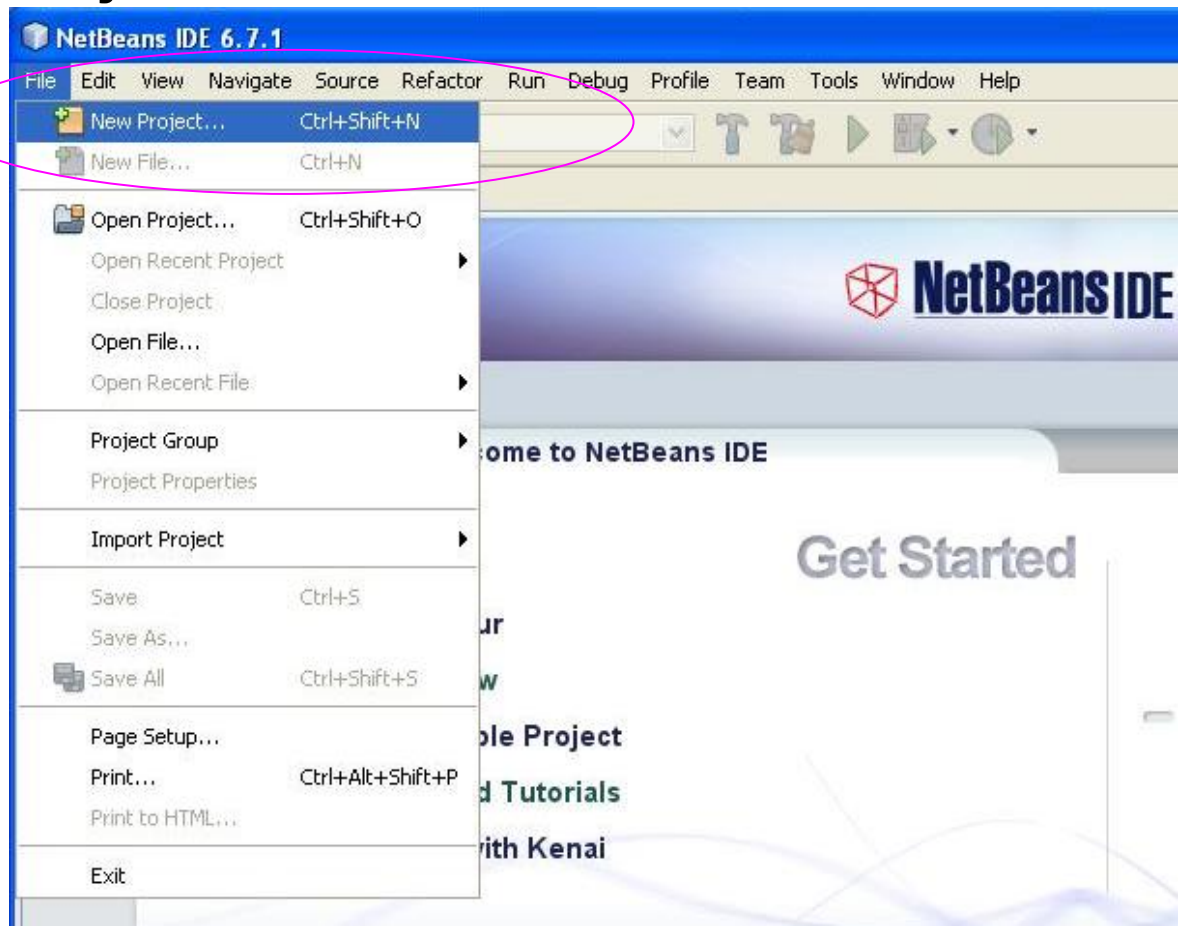
Demo. of Compilation + Interpretation…(Using Commands)

- Alternatively, teachers may consider to use free video capture tool such as “DEBUT Video Capture” program so as to produce a video file demonstrating the essential steps for issuing relevant commands for Java programs as HERE



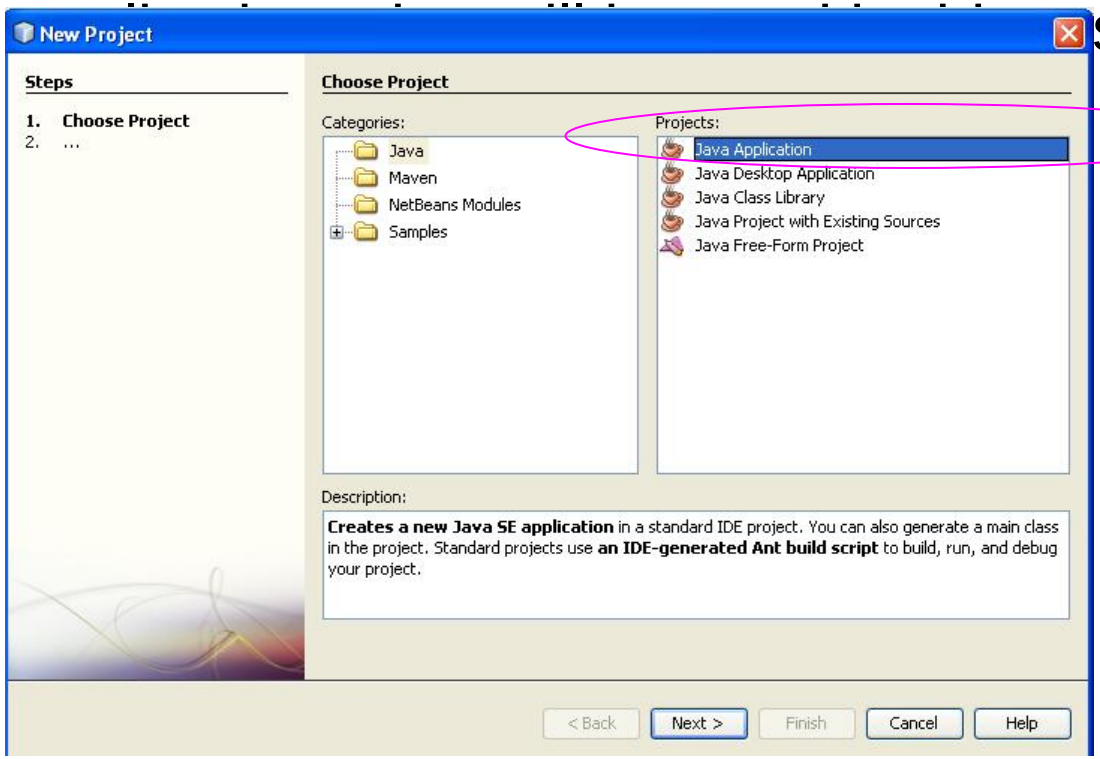
Console & Swing (or Windows) Applications in Java

- You can begin to write your first program by selecting **New Project** from the **File** menu as below.



Console & Swing (or Windows) Applications in Java

- The **New Project** dialog box will then be displayed for you to choose the type of applications to be developed. **Java [Console] Application** and **Java Desktop [Swing or Windows] Application** are the two types of source package.

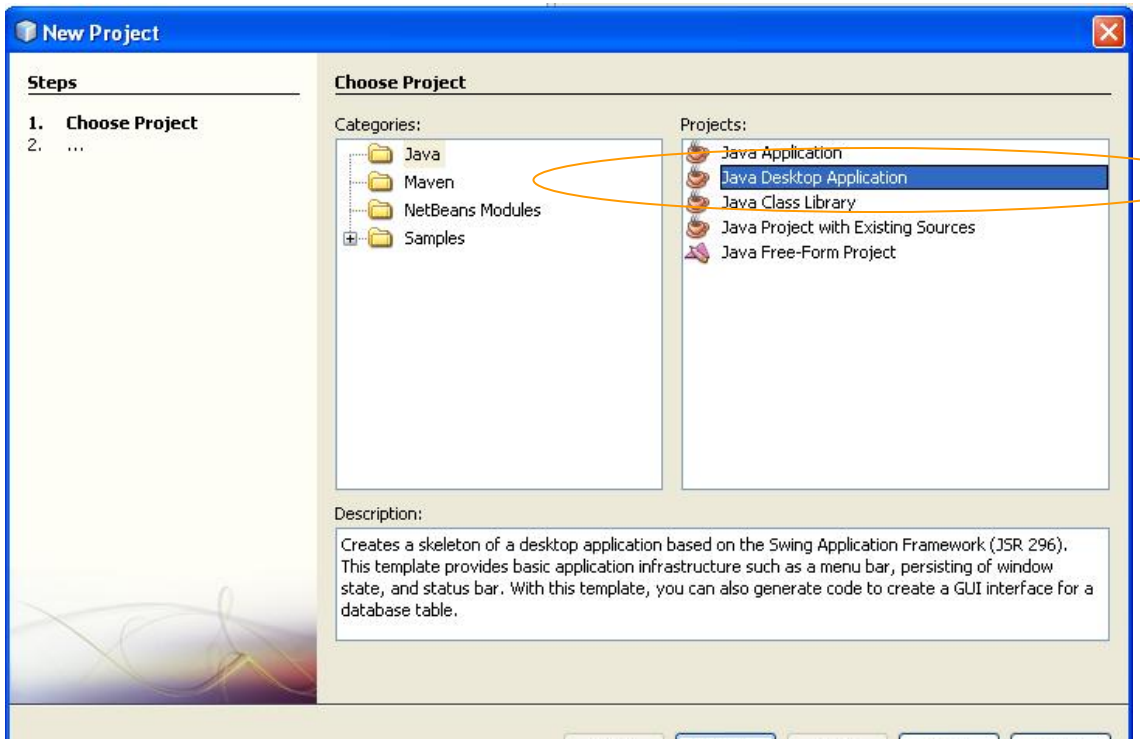


For Writing Java Console Application or Program, i.e. result will be generated in the "Output" Window as the normal console

(ref. Sec. 1.3 of Ch.1)

Console & Swing (or Windows) Applications in Java

- Alternatively, one can choose the **Java Desktop [Swing or Windows] Application** to produce window-based GUI and related GUI components & methods. However, this approach will be more complicated as it involves deeper understanding of the Swing library (of the Java Foundation Class [JFC]) under the Java2 platform.



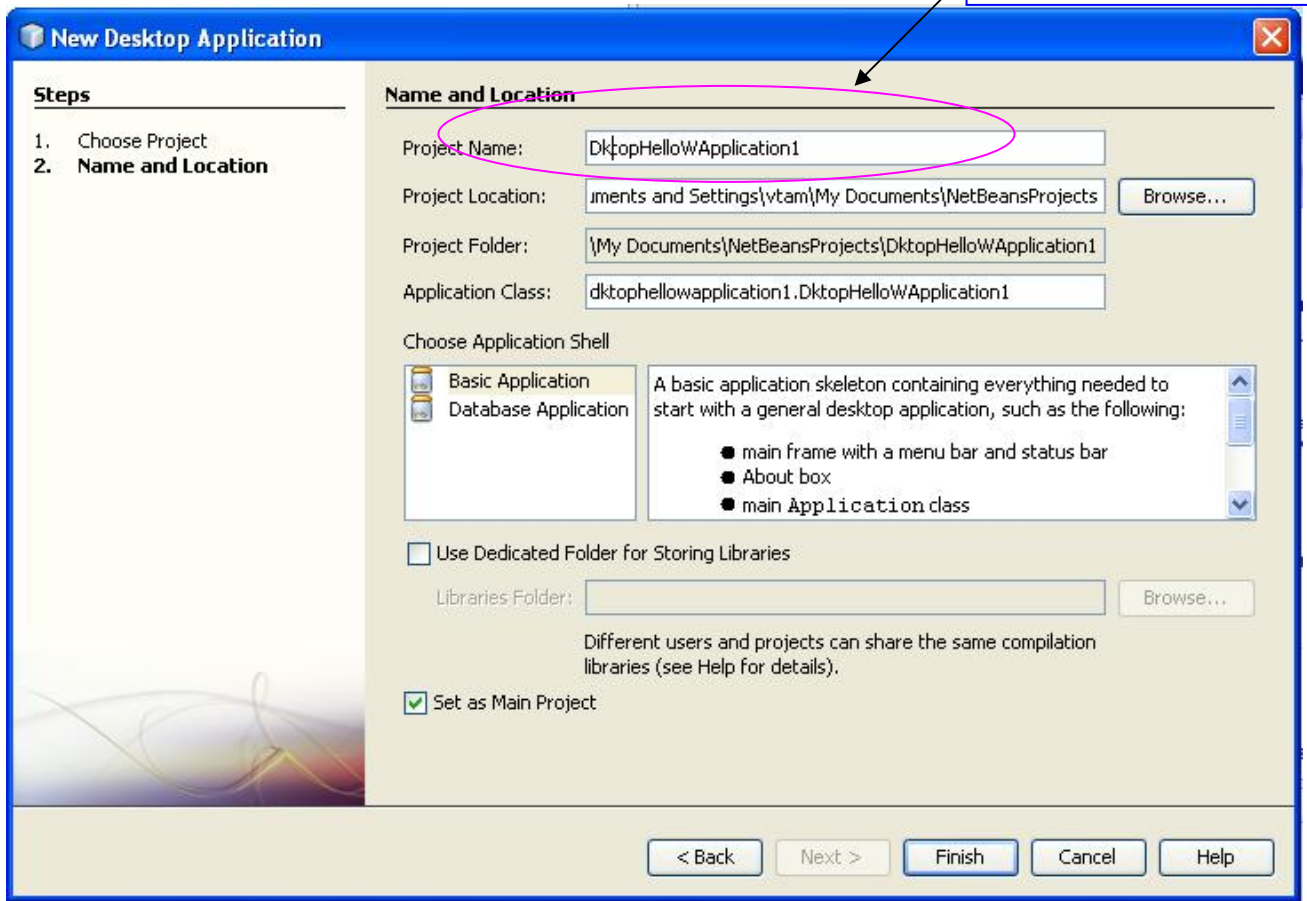
For Writing Java Swing Application or Program, i.e. result will be displayed in a separate window !!

(ref. Sec. 2 of Ch.1)

Console & Swing (or Windows) Applications in Java

- Here comes some snapshots of the steps involved for generating the **Java Desktop [Swing] Application** of the HelloWorld example covered in Chapter 1.

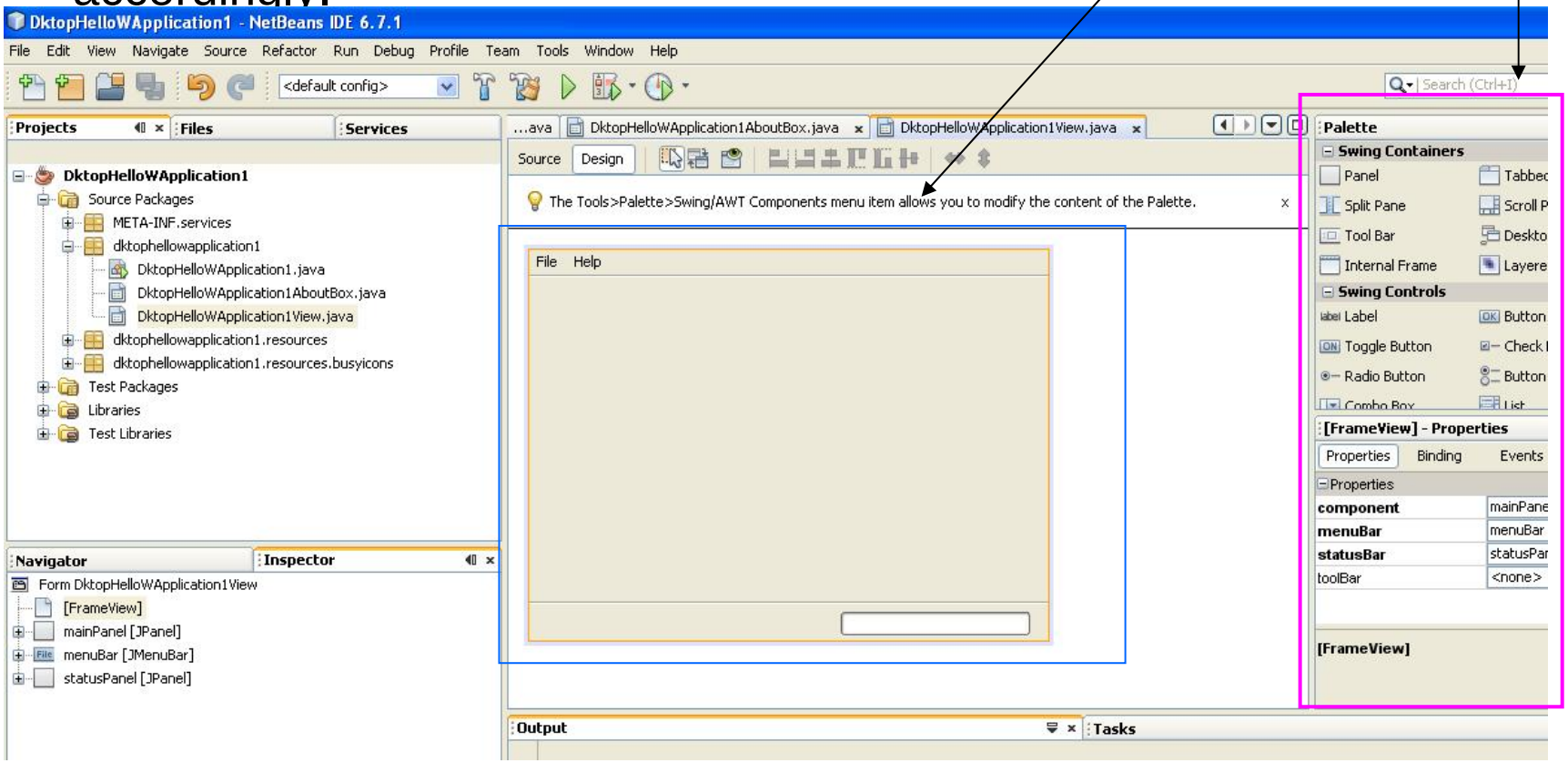
(1) Fill in the Project Name



Console & Swing (or Windows) Applications in Java

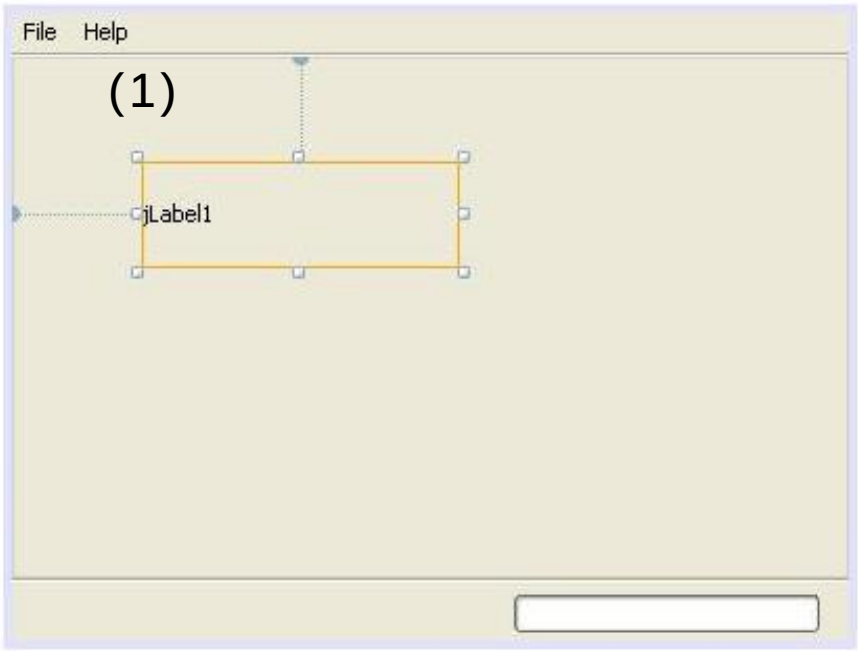
Palette Window

- Use the [Swing] Palette on the right-hand side to add/remove Swing components/controls into the generated **container window** in the center, and also set the properties of the GUI component accordingly.

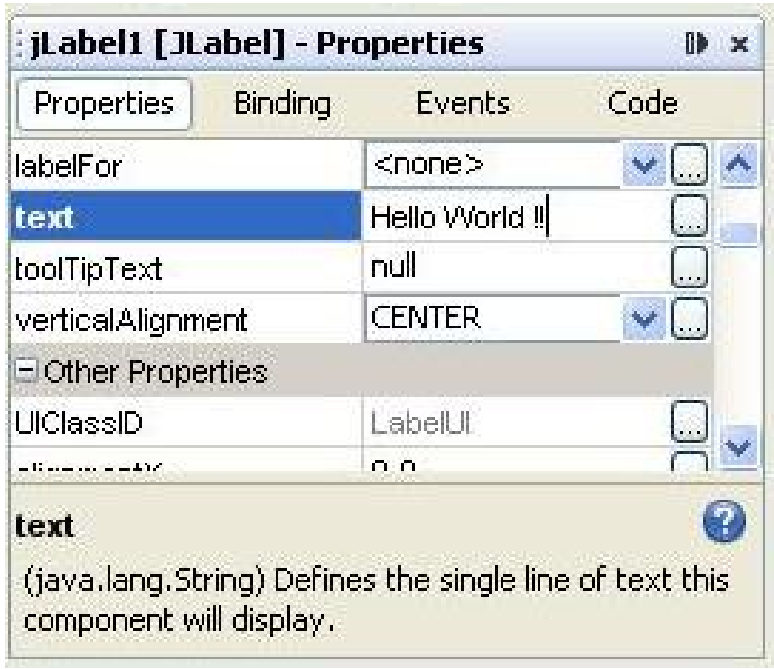


Console & Swing (or Windows) Applications in Java

- Add “Label” and set the text as “Hello World !!”.



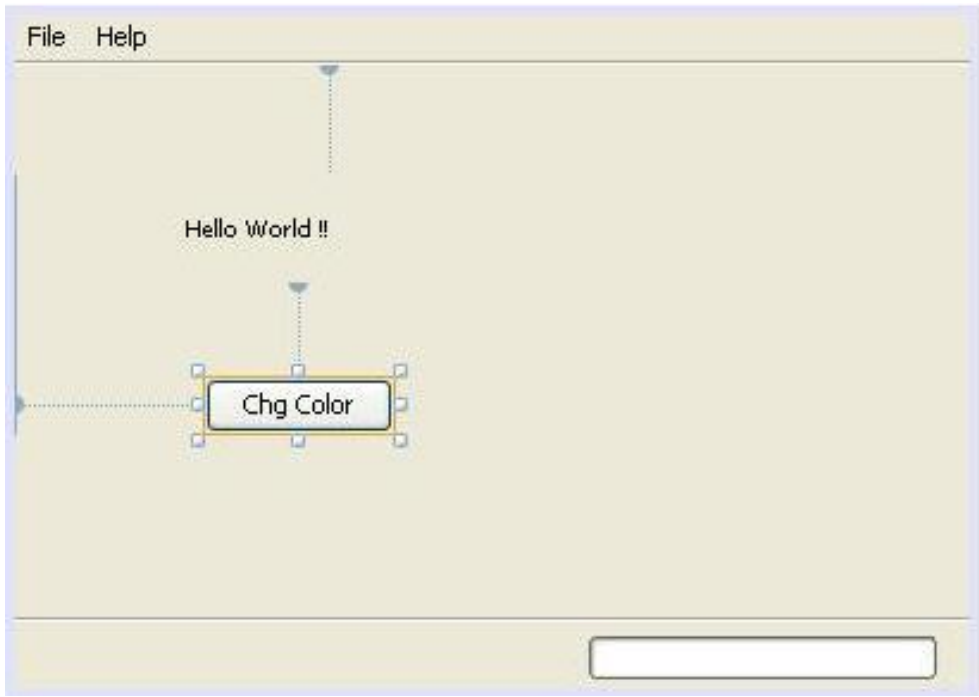
(2)



Console & Swing (or Windows) Applications in Java

- Add “Button” to change the background color.

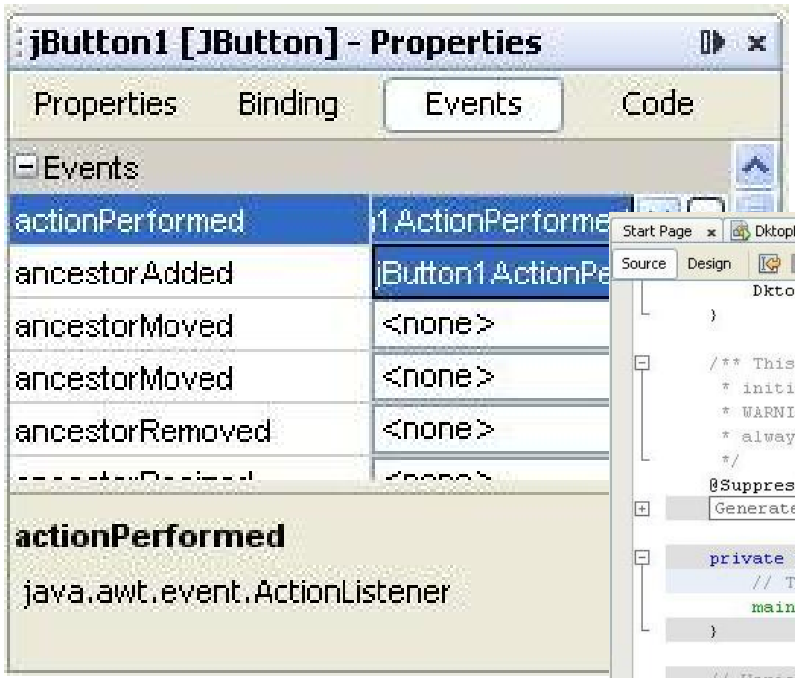
(3)



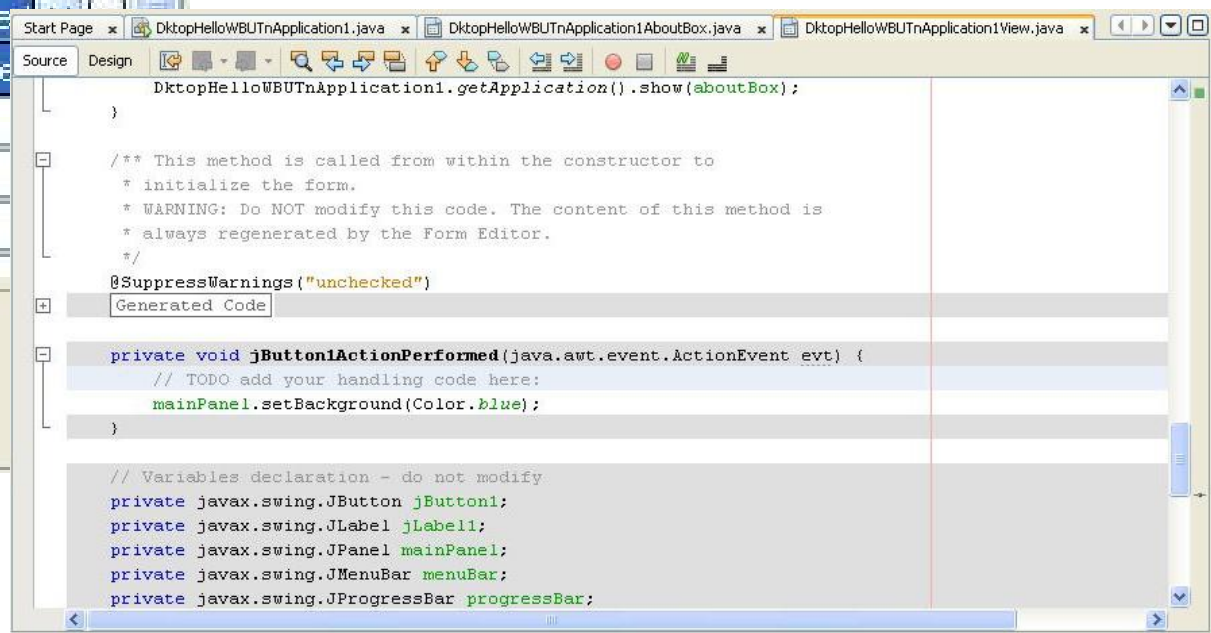
Console & Swing (or Windows) Applications in Java

- Set properties & write event codes for the “Button” object to change the background color.

(4)



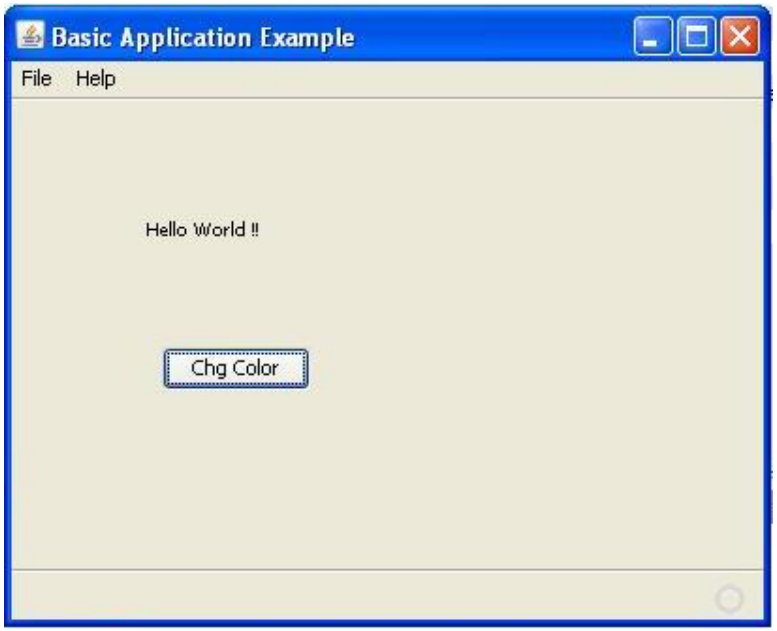
(5)



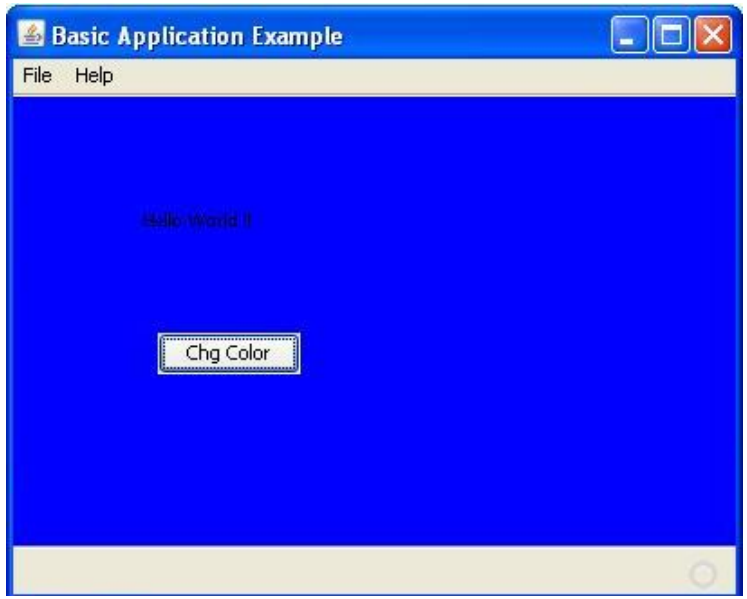
Console & Swing (or Windows) Applications in Java

- Click on the  on the toolbar, the following window (6) will be displayed. When you clicked on the “Chg Color” button, the background color of the window will be changed to Blue as (7).

(6)



(7)



OOP – Classes & Inheritance

- The major distinction between Java (as an OOP language) and imperative programming paradigms such as Visual Basic or C lies in
  - Classes and Objects;
  - Inheritance

>> mainly to facilitate software design & reuses

[ref. Chapter 6 – “Programming Modules in Java”]

OOP – Classes & Inheritance

- In procedural languages like C, Pascal or Visual Basic, the unit of programming is the function.
- However in Java, the unit of programming is the class from which *objects* are eventually instantiated (a technical term for 'created').
- Java classes contain attributes (implementing the class data) and methods (implementing the class behaviours). Clearly, the attributes and methods of a class/object are tied together as an inseparable programming unit in Java.

OOP – Basic Concepts of “Classes”

Real-World Applications::

Template/Model ↔ Actual Item

In Java,

Class ↔ Object

OOP & Inheritance in Java…

- The basic syntax to define a class in Java is given as follows.

```
class NewClass {
  . . .
} // end of class definition.
```

OOP & Inheritance in Java…

- By the class inheritance mechanism supported in Java, a newly created class of objects can be derived (or extended) by absorbing characteristics of existing classes and adding unique properties of their own.

E.g. an object of class 'Convertible' (to model convertible cars in the real world) will surely have the basic properties of the more general class 'Automobile', plus a convertible's roof that can go up or down for any convertible car.

OOP & Inheritance in Java…

- in Java, one can use the ***extends*** keyword, and followed by the name of the existing class to inherit from as below:

```
class Convertible extends Automobile {
  // new fields and methods defining a
     convertible car would go here
}
```

End
- Thank you

Q & A Session