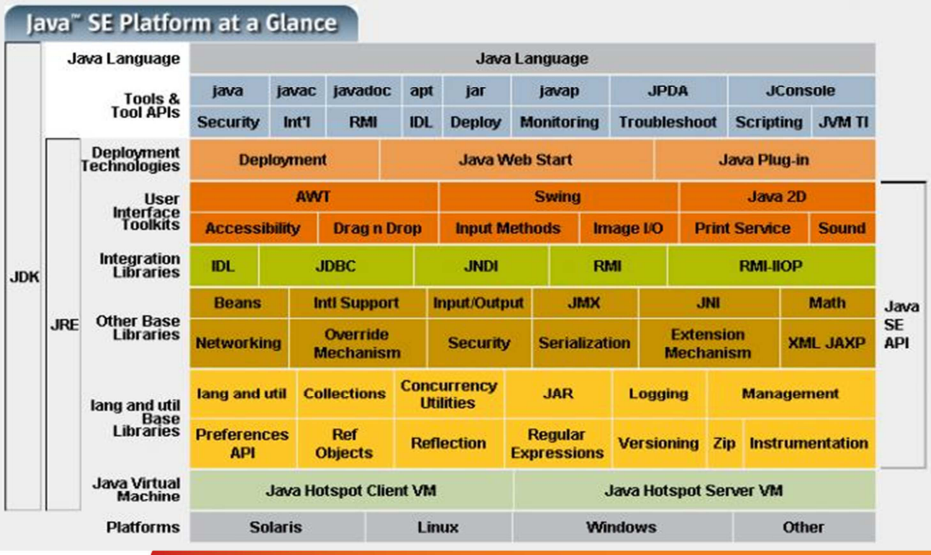
JAVA BASICS PART 1/2

Agenda

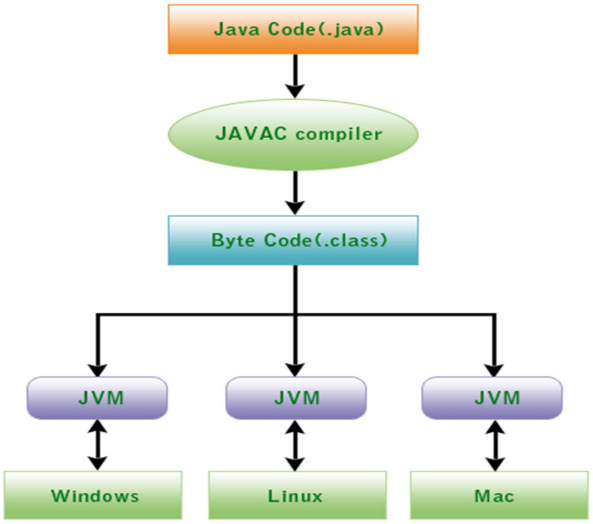
- Java development procedure
- Using Eclipse to develop Java project
- Java basic concept
- OOP with Java: Abstraction Encapsulation
- String literal
- System I/O
- String format
- DateTime structure object
- Coding Convention

JAVA DEVELOPMENT PROCEDURE

Java SE platform



Java development procedure



JAVA PROGRAM STRUCTURE

Java Program

```
package com.fpt.fwa.java;
/*
 * This is my first Java prj
 */
public class MyFirstJavaPrj {

    /**
     * This is a comment that will be used by javadoc tool
     * to generate the project document
     */

    // any program starts with Main method, with or without argument list
    public static void main(String[] args) {

        // Print out the string "This is my first Java prj" to project console
        System.out.println("This is my first Java prj");
    }
}
```


Present:

Comment

Structure

Statement

Keyword

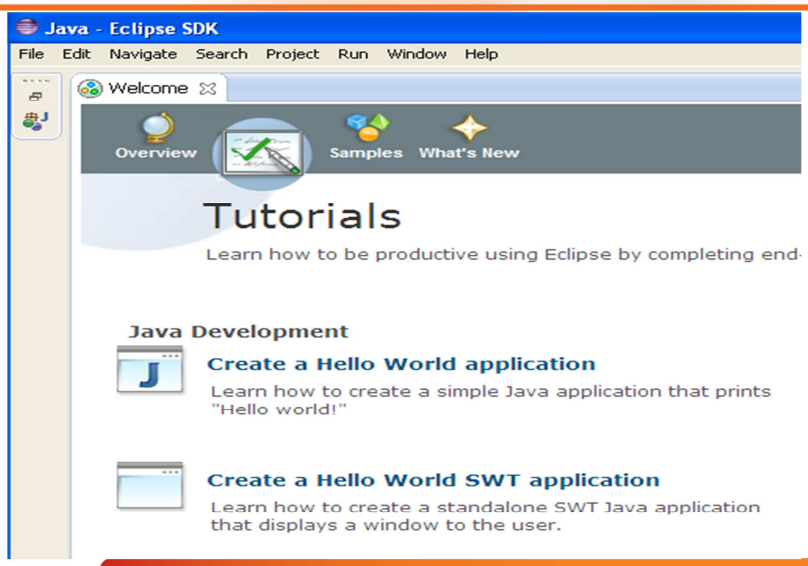
String constant/literal

USING ECLIPSE

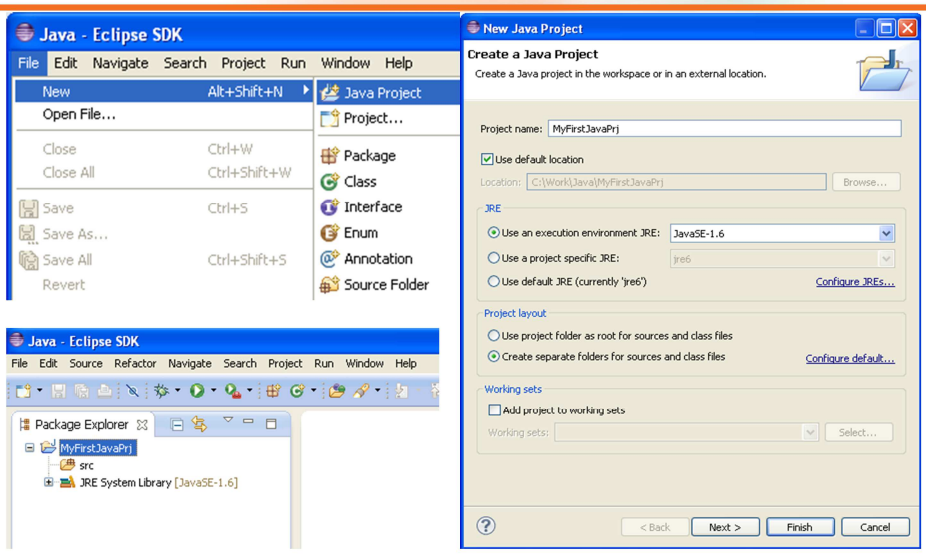
Agenda

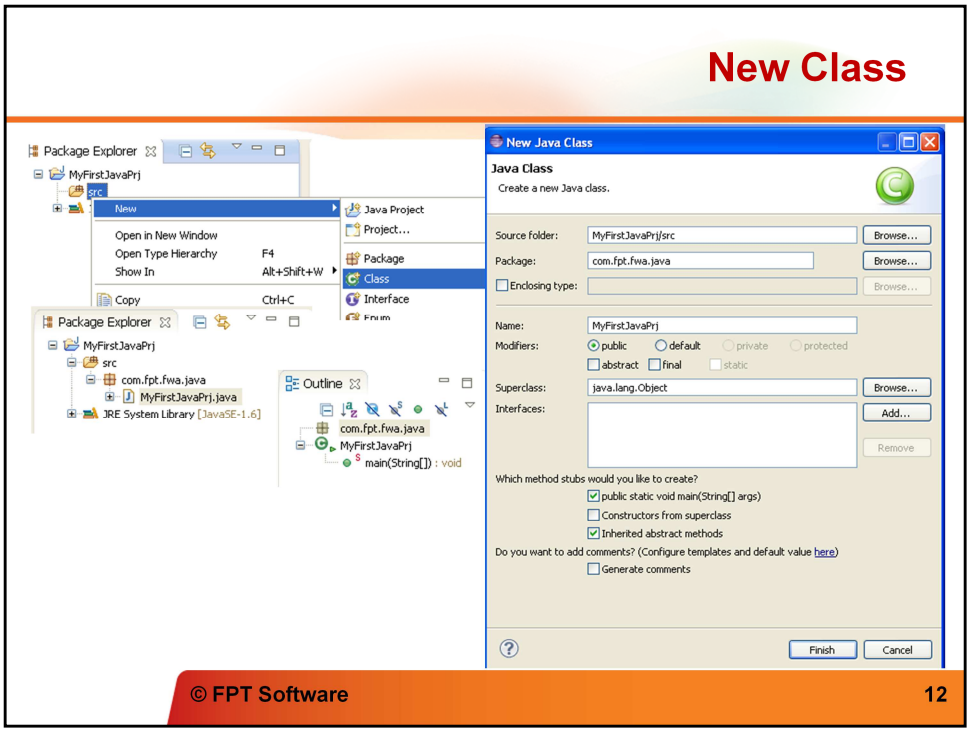
- Create a new Java console project
- Insert Java code into the new Java code file
- Run the new Java console project
- View the running result

Java Tutorial with Eclipse



New Java project





Present: Default structure, class file.

First project

MyFirstJavaPrj.java

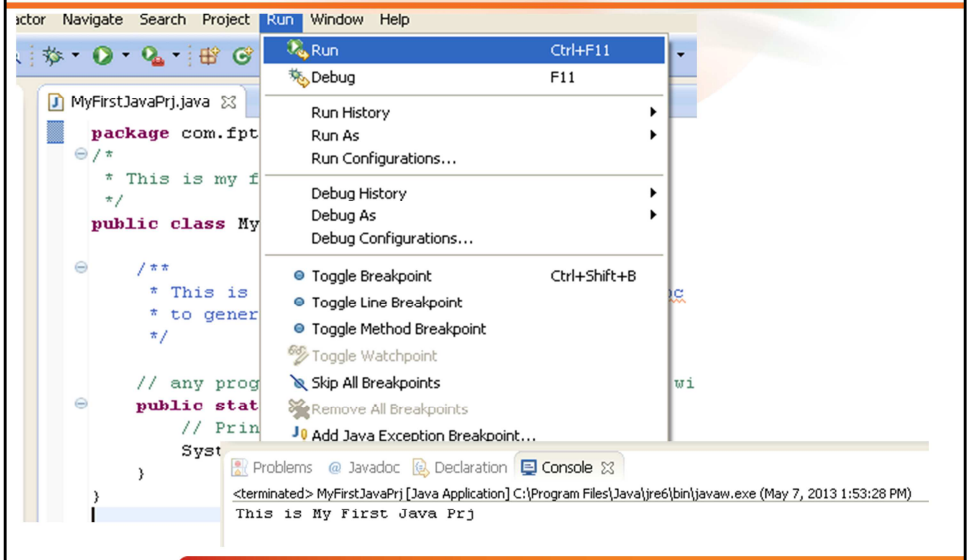
```
package com.fpt.fwa.java;
/*
 * This is my first Java prj
 */
public class MyFirstJavaPrj {

    /**
     * This is a comment that will be used by javadoc tool
     * to generate the project document
     */

    // any program starts with Main method, with or without argument list
    public static void main(String[] args) {

        // Print out the string "This is my first Java prj" to project console
        System.out.println("This is my first Java prj");
    }
}
```

Running/Debugging



Summary

- Create a new Java console project by the new project function
- Insert Java code into the new Java code file
- Run the new Java console project by debug function
- View the running result

JAVA BASIC

Agenda

- Key words
- Primitive types
- Variable: simple and array
- Operators
- Flow control statements
  - Branching statements
  - Iterate statements

Keywords

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

In red not mention in this course

Primitive types

- Integral – default 0: byte, short, int, long (0L)
- Default integral number type is int
- Decimal: float (0.0f), double (0.0d)
- Default floating point number type is double
- Boolean: boolean (false)
- Character: char (‘\0’)
- String: string (null)

Integral type has MAX_VALUE and MIN_VALUE => overflow exception or casting error

Memory capacity => Precision and range of each number type

Variable declaration and init

```
// Simple variable:
byte i;
// Many variables can have the same type declaration
float x, y, z;
int x = 1, y, z = 333_1500_24;  // underscore: Java7
string name – "new name";

// Array: Indexed by an integer from Zero
int[] x = {1, 2, 3}; char[] y = new char[6];  // OK
int x1 = x[0];                                // 1
int x[];                        // OK too but discouraged

// Multi dimension array/Array of array:
int[][] x = {{1, 2}, {3, 4}};
int[][] y = {{1, 2}, {3, 4, 5}};
int[][][] z = {{{1, 2}, {3, 4}}, {{5, 6}, {7, 8}}};
```

Number casting

```
// Smaller type number in memory size to
// wider one
byte b = 5;
int i = b;          // i = 5
// Wider type to smaller one
int i = 500;
byte j = i;         // compile error
byte j = (byte)i; // i = -12
// Range checked
byte i = (byte)(Byte.MAX_VALUE + 1); // -128
byte i = (byte)(Byte.MIN_VALUE - 1); // 127
```


Binary truncate

Aware the same effect in object casting

Operators

- References: . () [] new
- Arithmetic: + ++ - -- * / %
- Logical: & | ^ !
- Conditional: && (&) || ! == != > >= < <=
- Type verification: instanceof
- Bitwise: ~ >> >>> <<
- Assignment: = += -= *= /= %= &= |= ^= >>= <<=
- Selection: ?:

>>>: shift ignore sign – shift right with leftmost bit

>> : shift keep sign – shift right without leftmost bit

Flow control statements

- Branching
  - Selection: string x = 2 == 5 ? "Yes": "No";
  - if… else if … else
  - switch … case … default
- Iteration
  - for
  - do while
  - while
- Ignore and breaking
  - continue
  - break

Summary

- Key words: int, new, string, boolean…
- Primitive types: int, boolean, string…
- Variable: simple and array
- Operators: arithmetical, conditional…
- Branching statements: if, switch, selection
- Iterate statements: for, while, do while

METHOD

Method

```
// Signature: access modifier + return type
  +
// name + parameter list
public static boolean addUser(User anUser){
   return true;
} // Calling: name + argument list

if addUser(newUser){
   System.out.println("New user added");
} else {
   System.out.println("Adding user failed");
}
```

Undefined parameter number

```
// undefined parameter number
public void Multiply(int… list){};
Multiply();                          // OK
Multiply(1);                         // OK too
Multiply(1, 2, 3, 3, 4);             // still OK
Multiply(new int[] {1, 2, 3, 3, 4}); // OK

// predefined parameter number
public void Multiply(int[] list){};
Multiply();                         // Error
Multiply(1);                        // Error too
Multiply(new int[] {1, 2, 3, 3, 4}); // OK
```

OOP WITH JAVA PART I

Agenda

- Abstraction / Modelisation
- Encapsulation

OOP ABSTRACTION

Real life Object, Data and Action

- IT treats life's "object"
- Object has information – "data"
- "data" needs some "action" on its (treatments)

1: human, car, client, student

2: ": human sex, age, name, address.

3

Modelisation

- Modelisation: Object from real life to IT
  - Information
    - Wheel: 4 wheels
    - Main color: Orange
    - Rear port: 2 ports
    - With upper window: Yes
    - Seat: 2 seats
    - Cylinder volume:2.1L
  - Action
    - Engine start
    - Speed up, Slow down
    - Turn left, turn right
    - Stop

Abstraction/Modelisation

- Car is described by:
  - Number of wheels
  - Main color
  - Number of rear port
  - With upper window or not
  - Number of seats
  - Cylinder volume
  - Engine start action
  - Speed up action
  - Slow down action
  - Turn left action
  - Turn right action
  - Stop action

Each car has its out data and actions

Combine the action with data in an IT Object => OOP

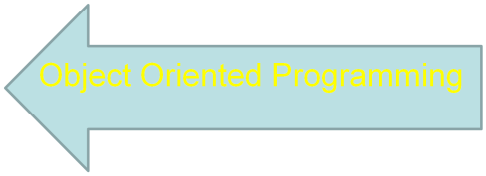
OOP ENCAPSULATION WITH JAVA

Agenda

- Class
- Member
- Constructor
- Access modifier
- Property
- Method overload
- Static class and static class features
- Static features: Class feature, not object's one

Encapsulation: Class, Member and Method

```
class Car{                      // Object model
   int numberWheels;            // Data => Member
   String mainColor;
   int numberRearPorts;
   boolean isWithUpperWindow;
   int numberSeats;
   float cylinderVolume;
   void engineStart(){…} // Action => Method
   void speedUp(){…}
   void slowDown(){…}
   void turnLeft(){…}
   void turnRight(){…}
   void stop(){…}
}
```



Instantiate – Constructor

```
// Instantiate - Create an object/"instance"
// from its model class with "default constructor"
Car aCar = new Car();
aCar = null;                  // now, aCar is no more an object
class Car{
   // Parameterized constructor
   Car(int NumberWheels,    String  MainColor,
       int NumberRearPorts, boolean isWithUpperWindow,
       int NumberSeats,     float   CylinderVolume){
       this.NumberWheels      = NumberWheels;
       this.MainColor         = MainColor;
       this.NumberRearPorts   = NumberRearPorts;
       this.isWithUpperWindow = isWithUpperWindow;
       this.NumberSeats       = NumberSeats;
       this.CylinderVolume    = CylinderVolume;
   }
   …
}
// New instantiation with parameterized constructor
Car aCar = new Car(2, "Orange", 2. true, 2, 2.1);
// all members of aCar are now called instance variables
// then, all methods are instance methods
```


Constructor by default

Not OK yet, see access modifier next page

Accessibility modifier

Modifier	Class	Package	Subclass	World
public	Y	Y	Y	Y
protected	Y	Y	Y	N
No modifier	Y	Y	Y	N
private	Y	N	N	N

Class: public or no modifier

Accessibility modifier: public/private

```
class Car{
   private int numberWheels; // private
   …
   public Car(…){…}     // public constructor
   public void engineStart(){…}
   public void speedUp(){…}
   public void slowDown(){…}
   public void turnLeft(){…}
   public void turnRight(){…}
   public void stop(){…}
}

Car aCar = new Car(…);
aCar.engineStart();     // OK, method is public
aCar.numberWheels = 6; // error, private member
```


Public data is denied

Summary

- Modelisation: IT object reflex real life object
- Encapsulation: Data and Action in one class
- Instance: a object of a class with specified data
- Instantiate: Create an object of a class
- Constructor: Activated at instance creation
- Access Modifier: Default, Private, Public…

STRING LITERAL

String literal

```
// Declaration and initialization
// "\" start an "escape" code
String s1 = "a\n", s2 = "\"", s3 = s2;

// assignment
s2 = s1;

// concatenation
s3 = s1 + s3;
```


Operation on String will be presented in next day

Escape code

\f	Formfeed	\"	Double quotation mark
\n	New line	\\	Backslash
\r	Carriage return	\ooo	ASCII character in octal notation
\t	Horizontal tab	\x hh	ASCII character in hexadecimal notation
\'	Single quotation mark	\x hhhh	Unicode in herxadeximal

SYSTEM I/O

Console I/O

```
// Write:
System.out.println(StrFormat, variableList);

// Read:
InputStreamReader sr – new
   InputStreamReader(System.in);
BufferedReader input = new BufferedReader(sr);
String line = input.readLine();
```


Dùng lớp Scanner không nói nhiều về luồng.

STRING FORMAT

String format

```
String header = String.format(
  "%1$-12s %2$8s %3$12s %2$8s %3$12s %4$14s\n",     // pattern string
  "City", "Year", "Population", "Change (%)");      // argument list

String body = String.format(
"%1$-12s %2$8tY %3$,12d %4$8tY %5$,12d %6$12.1 %%\n",
Name, BaseYear, BasePopulation, ObserveYear, ObservePopulation,
ObservePopulation - BasePopulation)/ (double) BasePopulation);

// Sample output
City             Year    Population     Year    Population     Change (%)
-----------------------------------------------------------------------
Los Angeles      1940     1,504,277     1950     1,970,358         31.0 %
New York         1940     7,454,995     1950     7,891,957          5.9 %
Chicago          1940     3,396,808     1950     3,620,962          6.6 %
Detroit          1940     1,623,452     1950     1,849,568         13.9 %
```


String format is used to replace the parameter into place holder with the needed format

Attention to header, argument used twice, first argument is left aligned

`BaseYear and Observed Year is in Date type`

Present the use of pattern

Number pattern in string format

'd'	decimal integer
'o'	octal integer
'x', 'X'	hexadecimal integer
'e', 'E'	decimal number in computerized scientific notation
'f'	decimal floating point number
'g', 'G'	computerized scientific notation or decimal format depending on the precision and the value after rounding.
'a', 'A'	hexadecimal floating-point number with a significant and an exponent

Number format directive list

Date time pattern in string format

't','T'	date and time format required prefix		'C'	Century in two digit format 00 - 99
'H'	Hour in 24-hour two digit format 00-23		'Y'	Year in four digit format 2013
'I'	Hour in 12-hour two digit format 00-12		'y'	year in two digit format 00 - 99.
'M'	Minute in two digit format 00 - 59		'j'	Day of year in three digit format 001 - 366
'S'	Seconds in two digit format 00 - 60		'm'	Month in two digit format 01 - 12.
'L'	Millisecond in three digit format 000 - 999.		'd'	Day of month in two digit format 01 - 31
'N'	Nanosecond in nine digit format 000000000 - 999999999.		'R'	"%tH:%tM"
'p'	morning or afternoon marker "am"/"pm"		'T'	"%tH:%tM:%tS".
'B','h'	Full month name, "January", "February".		'r'	"%tI:%tM:%tS %Tp"
'b'	Abbreviated month name, "Jan", "Feb".		'D'	"%tm/%td/%ty".
'A'	full name of the day in the week, "Sunday", "Monday"		'F'	"%tY-%tm-%td".
'a'	short name of the day in the week, "Sun", "Mon"		'c'	"%ta %tb %td %tT %tZ %tY"

Date and time string format directive

USING DATE AND TIME

Date and Time

```
import java.util.Date;
import java.text.SimpleDateFormat;

SimpleDateFormat df = new SimpleDateFormat(
   "yyyy-MM-dd hh:mm:ss.SSS");
try{
   Date d1 = df.parse("1940-01-01 15:28:31.976");
   Date d2 = df.parse("2014-13-36 36:65:82.976");
   String s = df.format(d2); // "2015-02-06 13:06:22.976"
} catch(ParseException e){
}
```

Date/Time pattern in SimpleDateFormat

y	Year	1996; 96
M	Month in year	July; Jul; 07
w	Week in year	27
W	Week in month	2
D	Day in year	189
d	Day in month	10
F	Day of week in month	2
E	Day in week	Tuesday; Tue
a	Am/pm marker	PM
H, h	Hour in day (0-23/0-12)	15, 12
k, K	H, h	24
m	Minute in hour	30
s	Second in minute	55
S	Millisecond	978
Z, Z	Full/Sort time zone	Pacific Standard Time; PST; GMT-08:00/-0800

Date and time string format directive

CODING CONVENTION

Coding convention in practice

```java
package com.fpt.fwa.java;
/*
 * This is my first Java prj
 */
public class MyFirstJavaPrj {

    /**
     * This is a comment that will be used by javadoc tool
     * to generate the project document
     */

    // any program starts with Main method, with or without argument list
    public static void main(String[] args) {

        // Print out the string "This is my first Java prj" to project console
        System.out.println("This is my first Java prj");
    }
}
```


Present:

Comment

Structure

Statement

Keyword

String constant/literal

Convention

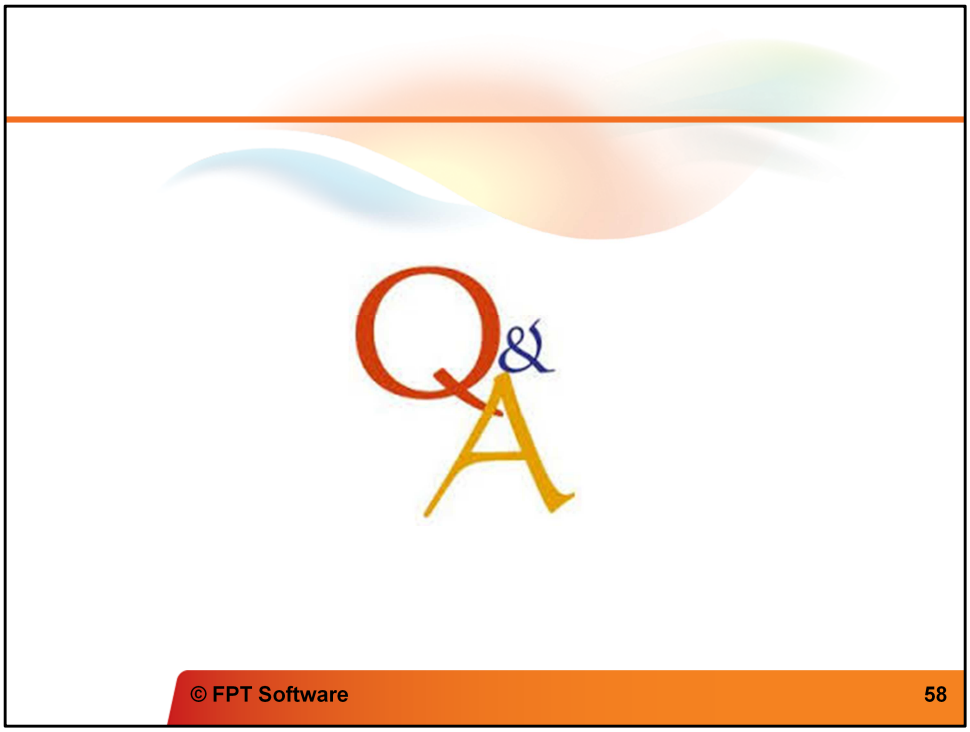
- Special case:
  - Index, simple variable – lowercase: x, y, i, a, b, c, name
  - Two letter name: UPPERCASE - IO, UI
  - Well know acronym: UPPERCASE – COM, XML…
  - Constant: UPPERCASE with underscore – MAX_AGE
  - Boolean status: isReady, isFinish
- Other – Camel style – lowercase then Uppercase
  - Ex: newUser, inputParameter

More coding convention

- FSOFT coding convention in QMS

Lesson summary

- Java development procedure: compile to bytecode
- Using Eclipse to create/run a new console project
- Java basic concept: types, statements, method
- OOP with Java: Abstraction and Encapsulation
- String literal: string in double quote
- System I/O: Console I/O
- String format: add variable value to string
- DateTime: Date time and SimpleDateFormat object
- Coding convention overview


Q&A
© FPT Software
58