Programming Concepts

2140101 Computer Programming for International Engineers

Objectives

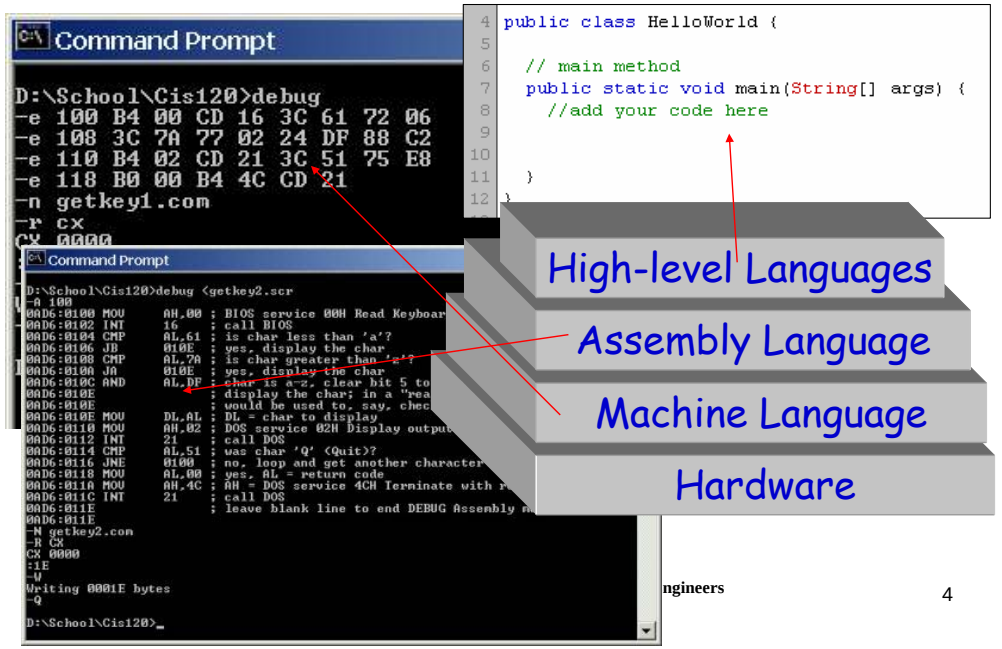
Students should:

- Know the steps required to create programs using a programming language and related terminology.
- Be familiar with the basic structure of a Java program.
- Be able to modify simple java to obtain desired results

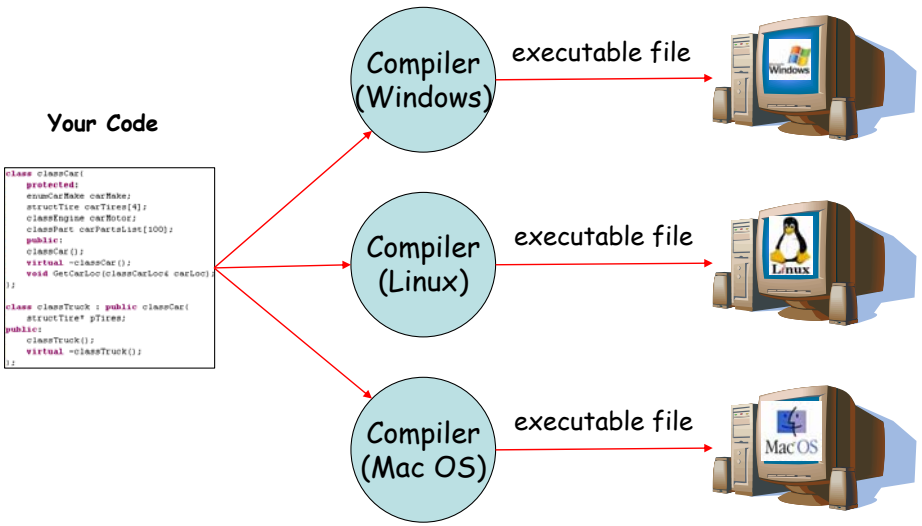
Programming Languages

- *Program* – a set or sequence of instructions that tell a computer what to do
- *Instructions* – described using programming languages

Categories

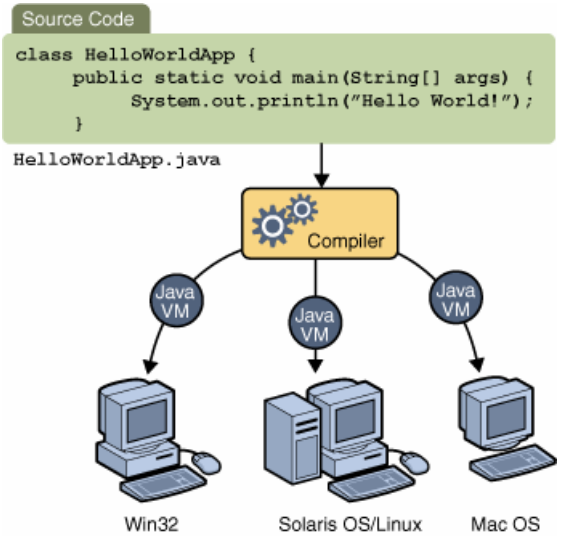


Traditional Compiled Programs

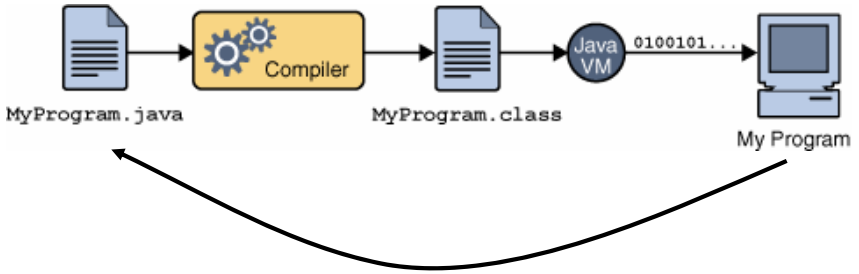


Running a Java Program

Write once, run everywhere



Programming Cycle



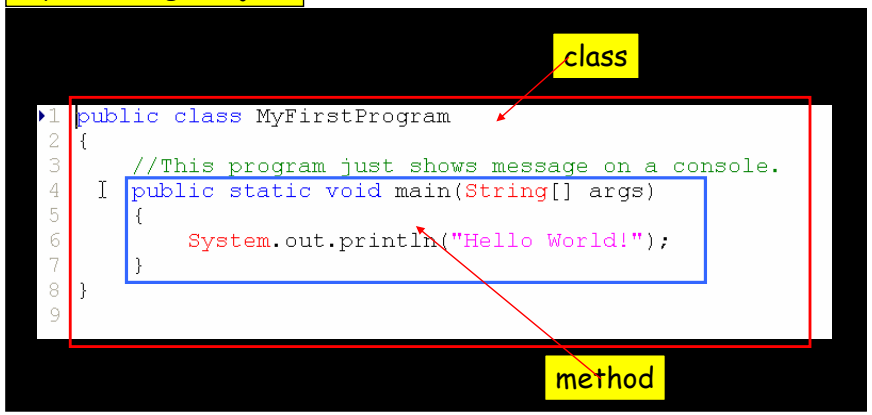
First Java Program

```
1  public class MyFirstProgram
2  {
3      //This program just shows message on a console.
4      public static void main(String[] args)
5      {
6          System.out.println("Hello World!");
7      }
8  }
9
```

```
C:\WINDOWS\system32\cmd.exe

D:\courses\2140101\JavaISE-2006-1\examples>javac MyFirstProgram.java

D:\courses\2140101\JavaISE-2006-1\examples>java MyFirstProgram
Hello World!

D:\courses\2140101\JavaISE-2006-1\examples>
```


Look Inside a Program

MyFirstProgram.java



Syntax, Keywords, & Identifiers

- *Syntax* – rules of the language, very strict
- *Keywords* – words that reserved for some purpose. You cannot use keywords as an identifier.
- *Identifiers* – names that given to classes, methods, and variables

Java Keywords

abstract	continue	for	new	switch
assert	default	goto*	package	synchronized
boolean	do	if	private	this
bit	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	volatide
const*	float	native	super	while

* not used

Comments

```
1  // An example of how to add comments
2  // This is a part of chapter 2 of 2140101 class notes
3  // Written by Atiwong Suchato
4
5  public class MyCommentedProgram
6  {
7      //Program starting point
8      public static void main(String[] args)
9      {
10         System.out.println("Hello World!");
11         // This part of code is commented out.
12         /*
13         System.out.print("Hello again.");
14         System.out.println();
15         */
16     }// end of main()
17 }// end of class
18
```


Readability

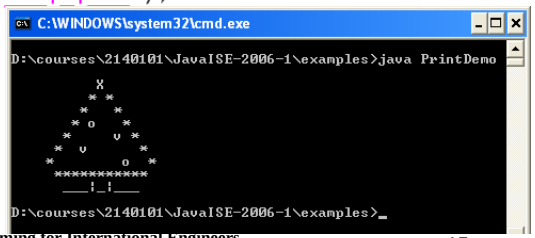
```
1 public class ExampleClass { public static void
2 main(String[] args){ double x=5; double z,
3 y=2; z=x+y;System.out.println(z);}}
```

```
1  public class ExampleClass
2  {
3  public static void main(String[] args)
4  {
5  double x = 5;
6  double z,y = 2;
7  z = x + y;
8  System.out.println(z);
9  }
10 }
```

```
1  public class ExampleClass
2  {
3      public static void main(String[] args)
4      {
5          double x = 5;
6          double z,y = 2;
7          z = x + y;
8          System.out.println(z);
9      }
10 }
```


println()

```
1  public class PrintDemo
2  {
3     public static void main(String [] args)
4     {
5        System.out.println("");
6        System.out.println("           X");
7        System.out.println("          * *");
8        System.out.println("         *   *");
9        System.out.println("        * o   *");
10       System.out.println("       *   v *");
11       System.out.println("      *  v     *");
12       System.out.println("     *         o *");
13       System.out.println("      ***********");
14       System.out.println("         ___|_|___");
15    }
16 }
17
```



println() and print()

```
1  public class PrintDemo2
2  {
3     public static void main(String [] args)
4     {
5        System.out.print("          ");
6        System.out.println(299);
7        System.out.println("+         800");
8        System.out.println("-------------");
9        System.out.print("          ");
10       System.out.println(299+800);
11       System.out.println("=============");
12    }
13 }
14
---------- Run ----------
          299
+         800
-------------
         1099
=============

Output completed (0 sec consumed) - Normal Termination
```

Escape Sequences

- \t – Tab
- \n – Newline
- \" – double quote
- \' – single quote
- \\ – backslash

```
1  public class EscapeSequence
2  {
3      public static void main(String[] args)
4      {
5          System.out.print("\tabc\td");
6          System.out.print("\n");
7          System.out.println();
8          System.out.println("\\ \" \' ");
9      }
10 }
```

```
---------- Run ----------
    abc d

\ " '
```


Variables

- Symbolic names of memory locations
- A *variable* must have name and data type associated with it.
- Must be declared before using it.

```
int x;
double y;
String myText;
```


Assign value to a variable

Assignment operator (=)

- Assign the value on the right to the variable on the left

```
x = 3;
Y = 6.5;
myText = “Java Programming”;

int z;
z = x;
```


Naming Rules for Java Identifier

- Cannot be a Java reserved word.
- Case-sensitive
- Unlimited length of sequence of Unicode letters and digits
- Must begin with a letter, underscore (_), or a dollar sign ($).
- White space not allowed

Naming Convention

- Use meaningful names
- For compound words use *camelCase*.
- Class names begin with an Uppercase letter.
  `Account, DictionaryItem, FileUtility, Article`
- Variable names and method names begin with a lowercase letter:
  `Height, speed, filename, tempInCelcius, imcomingMsg, textToShow`.
  (method names usually are verbs) `locate, sortItem, findMinValue, checkForError`
- For a constant use all uppercase letters and underscore (_) to separate words in compound names.
  `SOUND_SPEED, KM_PER_MILE, BLOCK_SIZE`

Expressions

- An *expression* is a value, a variable, a method, or one of their combinations that can be evaluated to a value.

```
3.875
a + b – 10
8 >= x
p || q
“go”
System.out.print(“go”)
Math.sqrt(2)
(x + 3 > y) && (x – 3 < z)
```


Statements

- Complete sentence that causes some action to occur.
- ends with a semicolon (;)

```
int k;
int j = 10;
double d1, d2, d3;
k = a + b – 10;
boolean p = (a >= b);
System.out.println(“go”);
sqareRootTwo = Math.sqrt(2);
```

- block of statements are the statements withing a block of curly braces ({ … })

```
{
    int m, n;
    m = 5;
    n = m * m;
}
```


Simple calculation

- Arithmetic operators
  - add (+)
  - subtract (-)
  - multiply (*)
  - divide (/)
  - modulo (%) : remainder after divide
- Assignment operator (=) is used to assign a value or the result from the calculation to a variable

A Simple Calculation Program

```
1  public class AverageDemo
2  {
3     public static void main(String[] args)
4     {
5        double avg, sum;
6        sum = 1.0 + 2.0 + 3.0 + 4.0 + 5.0 +
7              6.0 + 7.0 + 8.0 + 9.0 + 10.0;
8        avg = sum/10;
9        System.out.println(avg);
10    }
11 }
```

```
C:\WINDOWS\system32\cmd.exe
D:\courses\2140101\JavaISE-2006-1\examples>java AverageDemo
5.5

D:\courses\2140101\JavaISE-2006-1\examples>
```


Another Calculation

```
1  public class AverageDemo2
2  {
3     public static void main(String[] args)
4     {
5        int avg, sum;
6        sum = 1 + 2 + 3 + 4 + 5 +
7              6 + 7 + 8 + 9 + 10;
8        avg = sum/10;
9        System.out.println(avg);
10    }
11 }
```

```
C:\WINDOWS\system32\cmd.exe
D:\courses\2140101\JavaISE-2006-1\examples>java AverageDemo2
5

D:\courses\2140101\JavaISE-2006-1\examples>
```