Object Oriented Programming using Java 2

Java Basics

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Motivations for Java

Motivations for Java

- In-built garbage-collection facilities
- Portable facilities
- Multithreading concept
- Designed by James Gosling, Patrick Naughton, Chris Warth, Ed Frank and Mike Sheridan at Sun Microsystems in 1991

Features of Java

- Java is a pure object-oriented language
- Java is completely independent of any particular o.s. platform. As long as the JVM (Java Virtual Machine) is installed, the java program will run identically on all machines
- A java program is compiled & interpreted. The source code is compiled into an intermediate byte code file. This intermediate file is interpreted by a Java interpreter
  - Byte code makes the java program portable across different h/w and o.s. platforms
  - Increased security due to the control of the JVM over the execution of the byte code file

Features of Java...

- Java provides implicit support to memory management. The automatic technique of freeing unused memory is called garbage collection
- Java has a strong exception-handling mechanism
- Java supports multithreading, i.e. a single process can do multiple tasks apparently simultaneously
- Java supports TCP/IP and UDP protocol families and it can be used for programming in a networked environment

Object-Oriented Programming

Object-Oriented Programming

- The data is treated as the most important element and it can't flow freely around the system
- **Encapsulation**
  - Class is used as a unit to group related attributes and operations together. The outside world can interact with the data stored in the variables that represent the attributes of the class only through the operations of that class
- **Abstraction & Implementation Hiding**
  - Class is one abstract unit. In order to perform a task that involves an object of that class, a message must be sent to the object asking it to execute the respective operation
  - Implementation hiding means the manner in which data is stored in an object, is hidden from the outside

Object-Oriented Programming...

Object-Oriented Programming...

- **Inheritance, Dynamic Binding and Polymorphism**
  - Inheritance is a property by which one class inherits the features of another class
  - Dynamic binding or runtime binding or late binding is a technique in which the piece of code to be executed is determined only at runtime
  - Polymorphism means the ability to take more than one form
- **Overriding and Overloading**
  - Overriding of a method in a class is the redefinition of a method in the sub-classes of that class
  - Method overloading is a term used when several methods within the same class have the same name but different signatures (signature means the number and type of parameters in the method)
  - *In overriding, the same method name and same signature can be used in different sub-classes of the class and defined differently in each sub-class; whereas in overloading, the same name of the method with different signatures is used multiple times in the same class*

Advantages of Object-Oriented Programming

Advantages of Object-Oriented Programming

- It supports reusability of code
- Frameworks can be created
- It provides a clear modular structure for programs
- Easier for updating of codes in methods

Java Language Features

Java Language Features

- **Simple**: Java is designed to be easy for the professional programmer to learn and use
- **Object-oriented**: a clean, usable, pragmatic approach to objects, not restricted by the need for compatibility with other languages
- **Robust**: restricts the programmer to find the mistakes early, performs compile-time (strong typing) and run-time (exception-handling) checks, manages memory automatically
- **Multithreaded**: supports multi-threaded programming for writing program that perform concurrent computations
- **Architecture-neutral**: JVM provides a platform-independent environment for the execution of byte code

Java Language Features...

Java Language Features...

- **Interpreted and high-performance**: Java programs are compiled into an intermediate representation- byte code:
  - can be later interpreted by any JVM
  - can be also translated into the native machine code for efficiency
- **Distributed**: Java handles TCP/IP protocols, accessing a resource through its URL much like accessing a local file
- **Dynamic**: substantial amounts of run-time type information to verify and resolve access to objects at run-time
- **Secure**: programs are confined to the Java execution environment and can't access other parts of the computer

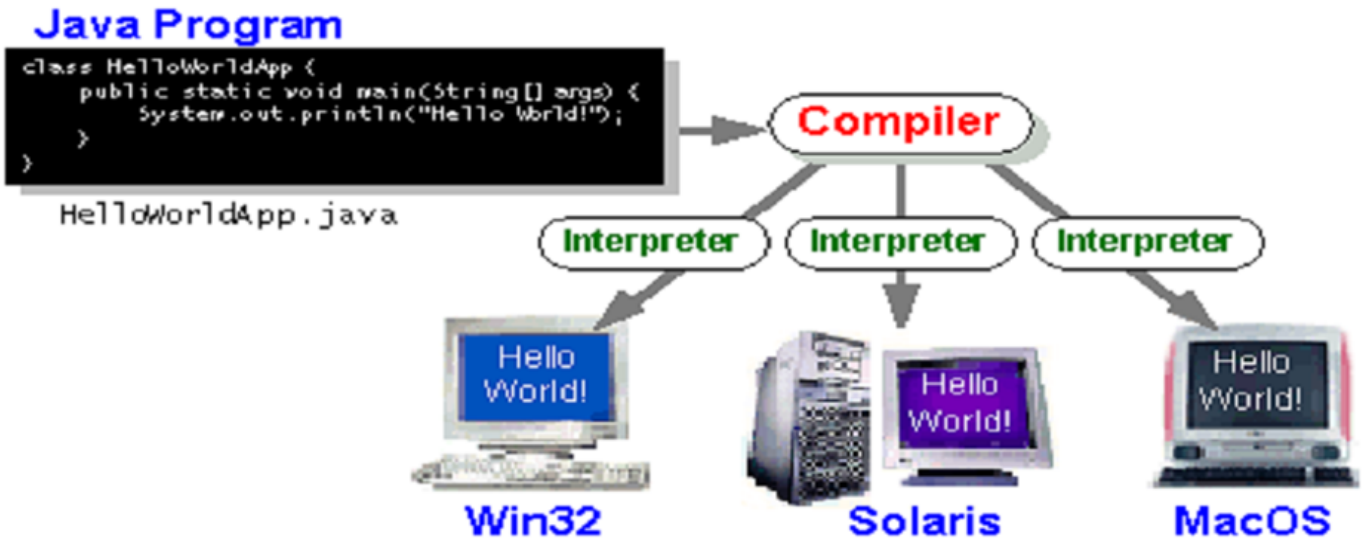
JDK, JRE & JVM

- **JVM** (Java Virtual Machine) is an abstract machine. It is a specification that provides runtime environment in which java bytecode can be executed. *JVM is platform independent*
  - Loads code
  - Verifies code
  - Executes code
  - Provides runtime environment
- **JRE** (Java Runtime Environment) is used to provide runtime environment. It is the implementation of JVM. It contains set of libraries + other files that JVM uses at runtime. *JRE is platform dependent*
- **JDK** (Java Development Kit) contains JRE + Development tools. *JDK is platform dependent*

Tools in the JDK

- JDK contains the necessary facilities for the compilation of Java programs into intermediate byte code and their interpretation. It consists of various tools that can be used by the programmer to develop java programs:
  - **javac**: it takes as input a java source code file and produces a class file that contains the byte code
  - **java**: it takes as input, a class file containing the byte code and runs the program by interpreting it
  - **jdb**: it is a debugger that assists the developer in detecting errors
  - **javadoc**: it takes a java source code file as input and produces documentation in html for it
  - **javah**: it takes a java source code file as input and produces header files for use with native methods
  - **javap**: it takes the byte code file as input and produces a file that gives a description of the source code file from which the byte code file was created after compilation. It is a java disassembler
  - **appletviewer**: it is a tool that permits the user to execute java applets without using any java-compatible browser

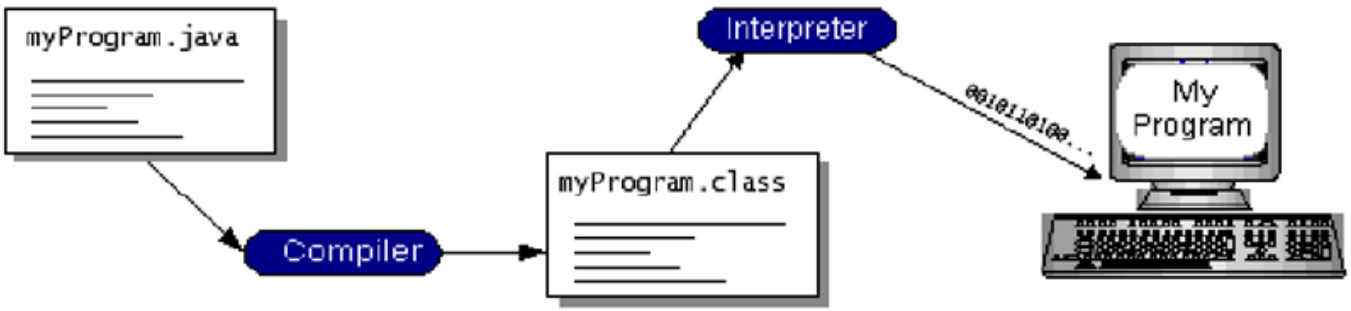
Java Platform Independence



Java Program Execution

Java Program Execution

- Java programs are both compiled and interpreted
- Steps:
  - write the Java program
  - compile the program into byte code
  - execute (interpret) the byte code on the computer through the Java Virtual Machine
- *Compilation happens once, while Interpretation occurs each time the program is executed*



Simple Java Program

Simple Java Program

A class to display a simple message:

```
class  MyProgram {
        public static void main(String[] args) {
                System.out.println("First Java program");
                }
        }
```

- MyProgram is the name of the class
- main is the method with one parameter args and no results
- println is a method whose purpose is to print the argument passed to it to the console and move the cursor to the next line; in the standard System class
- out refers to the standard output stream

Simple Java Program...

Simple Java Program...

- A class is the basic building block of Java programs
- A class encapsulates:
  - data (attributes) and
  - operations on this data (methods)

  and permits to create objects as its instances
- The main method must be present in every Java application
- public static void main(String[] args) where:
  - public means that the method can be called by any object
  - static means that the method is shared by all instances. It is a class method
  - void means that the method does not return any value
- When the interpreter executes an application, it starts by calling its main method which in turn invokes other methods in this or other classes
- The main method accepts a single argument- a string array, which holds all command-line parameters

Primitive Data types

Primitive Data types

- byte: is a signed 8-bit integer
  - byte b;
- short: is a signed 16-bit integer
  - short s;
- int: is a signed 32-bit integer
  - int i;
- long: is a signed 64-bit integer
  - long l;
- float: is a single-precision 32-bit floating point number
  - float f;
- double: is a double-precision 64-bit floating point number
  - double d;
- char: is a single 16-bit Unicode character
  - char c;
- boolean: is a data type having only 2 values: true and false
  - boolean b;

Primitive Data types...

Primitive Data types...

- Java is a strongly-typed language
- Every variable and expression has a type
- Every type is strictly defined
- All assignments are checked for type-compatibility
- No automatic conversion of non-compatible, conflicting types
- Java compiler type-checks all expressions and parameters
- Any typing errors must be corrected for compilation to succeed

Elements of Java Program

Elements of Java Program

- Whitespaces: is a space, or tab or a newline character
  - class HelloWorld
- Identifiers: used to identify variables, methods and classes. They can be a sequence of alphabets, numbers, underscore character and a dollar-sign character. It can't begin with a digit
  - Identifiers are case- sensitive
  - int e, E;
- Literals: used to specify constant values in a java program
  - 100
  - 300.56
  - 'M'
  - Ïndia"

Elements of Java Program...

- Comments:
  - Single-line Comments:
    //end of the loop

    /*end of the loop*/
  - Multiline Comments:
    /*this is a multiline comments
    of two lines*/
  - Documentation Comments:
    /** documentation */
- Separators:
  - Semicolon (;)
  - Period (.)
  - Comma (,)
  - Brackets ([ ])
  - Curley braces ({ })
  - Parentheses (( ))

Elements of Java Program...

Keywords: reserved words in java:

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

Elements of Java Program...

- Escape sequences:
  - **ddd**: octal character ddd
  - **uxxxx**: hexadecimal Unicode character xxxx
  - \': single quote
  - \": double quote
  - \ \: backslash
  - **r**: carriage return
  - **n**: new line
  - **f**: form feed
  - **t**: tab
  - **b**: backspace

Variables in Java

- All variables must be declared before they can be used
  - ***datatype identifier [=value];***
  - char c='A';
- It is also possible to initialize a variable dynamically at runtime
  - double area= length * breadth;
- Scope and Lifetime of Variables:
  - Scope determines the visibility of program elements with respect to other program elements
  - Block defines a scope. A block starts with a '{' and ends with a '}'
  - Each time a new block is started, a new scope begins
  - The life of the variable is within its scope

Automatic conversion of compatible data types

Automatic conversion of compatible data types

- If the data type of the value and the data type of the variable are compatible and the data type of the variable is larger than that of the value, then java will automatically allow the conversion
- byte $\rightarrow$ short, int, long, float, double
- short $\rightarrow$ int, long, float, double
- int $\rightarrow$ long, float, double
- long $\rightarrow$ float, double
- float $\rightarrow$ double
- char $\rightarrow$ int, long, float, double

Type Casting

Type Casting

- If the data type of the value is larger than the data type of the variable to which it is being assigned, then cast is used to explicitly convert the data type of the value to the data type of the variable
  - **(targetType) value**
  - double d=12.34D;
  - float f=(float) d;
- *If the whole number is too large to fit into the target integer type, the value will be reduced modulo the target type's range*

Promotion of data types in expressions

- byte and short are always promoted to int
- if one operand is long or float or double, the whole expression is promoted to long or float or double respectively

Operators

- Operators are used to build value expressions
  - Unary
  - Binary
  - Ternary
- **Arithmetic** Operator
  - +, -, *, /, %
- **Relational** Operator
  - Outcome is always a value of type Boolean
  - ==, $\neq$, < , $\leq$, > , $\geq$
- **Logical** Operator
  - Logical operators act upon Boolean operands only
  - &, |, !, $\wedge$
- **Short Circuit Logical** Operator
  - If these operators are used, java will not evaluate the second operand if the result can be determined by the first operand alone
  - & &, ||

Operators...

- **Increment** & **Decrement** Operator
  - The operand must be a numerical variable
  - prefix version evaluates the value of the operand after performing the increment/decrement operation
  - postfix version evaluates the value of the operand before performing the increment/decrement operation
- **Bitwise Logical** Operator
  - &, |, ∼, ∧, < <, > >, > > >
- **Assignment** Operator
  - Types of the variable and expression must be compatible
  - =
- Other Operators
  - Conditional operator (? :)
  - ⊏ ⊐
  - .
  - (params)
  - (type)
  - new
  - instanceof

Operator Precedence

Operator Precedence

When operators have the same precedence, the earlier one binds stronger

highest			
()	[]	.	
++	--	~	!
*	/	%	
+	-		
>>	>>>	<<	
>	>=	<	<=
==	!=		
&			
^			
|			
&&			
||			
?:			
=	op=		
lowest			

Selection Statements

Selection Statements

- Java selection statements allow to control the flow of program's execution based upon conditions known only during run-time
- if statement

```
if (expression)
        {
        statement
        }
```

  - The expression must be of type Boolean

Selection Statements...

Q: Input a number and check whether it is positive

```
import java.io.*;
class demo
{
public static void main(String []args)
    {
    String s=null;
    BufferedReader br=new BufferedReader(new
     InputStreamReader(System.in));
     System.out.println("Enter any number:");
     try
            {
            s=br.readLine();
            }
     catch (Exception e){}
     int i=Integer.parseInt(s);
     if(i>0)
            {
            System.out.println("The number "+i+" is
            positive");
            }
}
}
```

```
import java.util.*;
class demo
{
public static void main(String []args)
{
System.out.println("Enter any number:");
Scanner sc=new Scanner (System.in);
int i=sc.nextInt();
if(i>0)
            {
             System.out.println("The number "+i+" is
             positive");
            }
}
}
```

Selection Statements...

Selection Statements...

- if-else statement

```
if (expression) {
                    statement1
                    }
          else
                    {
                    statement2
                    }
```

- if-else-if statement

```
if (expression1) statement1
    else if (expression2) statement2
    else if (expression3) statement3
    ...
    else statement
```

Switch Statement

- switch provides a better alternative than if-else-if when the execution follows several branches depending on the value of an expression

```
switch (expression) {
        case value1: statement1; break;
        case value2: statement2; break;
        ...
        default: statement;
        }
```

- *Expression must be of type byte, short, int or char*
- *Each of the case values must be a literal of the compatible type*
- *Case values must be unique*
- *Break makes sure that only the matching statement is executed*

Iteration Statements

Iteration Statements

- Java iteration statements enable repeated execution of part of a program until a certain termination condition becomes true
- while statement

```
while (expression)
        statement
```

- do-while statement

```
do
        statement
while (expression);
```

- for statement

```
for (initialization; termination; increment)
        statement
```

Jump Statements

- Java jump statements enable transfer of control to other parts of program
- break statement
  - *break;*
  - Java does not have goto statement
  - *break label;*
  - *label: { ... }*
- continue statement
  - The break statement terminates the block of code, in particular it terminates the execution of an iterative statement
  - The continue statement forces the early termination of the current iteration to begin immediately the next iteration
  - *continue;*
  - *continue label;*

Math Class

- Math class contains all the floating-point functions that are used for geometry and trigonometry
  - Math.sin()
  - Math.sinh()
  - Math.cbrt(), Math.exp(), Math.log(), Math.log10(), Math.pow(), Math.sqrt()
  - Math.abs(), Math.ceil(), Math.floor()
  - Math.max(), Math.min(), Math.round()
  - Math.random()
  - Math.toDegrees(), Math.toRadians()
  - Math.PI