Programming Language II
CSE-215

Prof. Dr. Mohammad Abu Yousuf
yousuf@juniv.edu

Overview of Java

What is java?

- Developed by Sun Microsystems (James Gosling)
- A general-purpose object-oriented language
- Based on C/C++
- Designed for easy Web/Internet applications
- Widespread acceptance

Java Features (1)

- **Simple**
  - fixes some clumsy features of C++
  - no pointers
  - automatic garbage collection
  - rich pre-defined class library http://java.sun.com/j2se/1.4.2/docs/api/
- **Object oriented**
  - focus on the data (objects) and methods manipulating the data
  - all functions are associated with objects
  - almost all datatypes are objects (files, strings, etc.)
  - potentially better code organization and reuse

Java Features (2)

- **Interpreted**
  - java compiler generate byte-codes, not native machine code
  - the compiled byte-codes are platform-independent
  - java bytecodes are translated on the fly to machine readable instructions in runtime (Java Virtual Machine)
- **Portable**
  - same application runs on all platforms
  - the sizes of the primitive data types are always the same
  - the libraries define portable interfaces

Java Features (3)

- **Reliable**
  - extensive compile-time and runtime error checking
  - no pointers but real arrays. Memory corruptions or unauthorized memory accesses are impossible
  - automatic garbage collection tracks objects usage over time
- **Secure**
  - usage in networked environments requires more security
  - memory allocation model is a major defense
  - access restrictions are forced (private, public)

Java Features (4)

- **Multithreaded**
  - multiple concurrent threads of executions can run simultaneously
  - utilizes a sophisticated set of synchronization primitives (based on monitors and condition variables paradigm) to achieve this
- **Dynamic**
  - java is designed to adapt to evolving environment
  - libraries can freely add new methods and instance variables without any effect on their clients
  - interfaces promote flexibility and reusability in code by specifying a set of methods an object can perform, but leaves open how these methods should be implemented
  - can check the class type in runtime

Java Disadvantages

- **Slower than compiled language such as C**
  - an experiment in 1999 showed that Java was 3 or 4 times slower than C or C++

    title of the article: "Comparing Java vs. C/C++ Efficiency Issues to Interpersonal Issues" (Lutz Prechelt)
  - adequate for all but the most time-intensive programs

Introduction

- Java is an object-oriented programming language. Everything in Java is an object. Object-oriented means we organize our software as a combination of different types of objects that incorporates both data and behavior.
- Java includes the three pillars of object-oriented development:
  - **Encapsulation**
  - **Inheritance**
  - **Polymorphism**

Encapsulation

- ***Encapsulation is the mechanism that binds together code and the data it manipulates, and keeps both safe from outside interference and misuse.***
- In an object-oriented language, code and data may be combined in such a way that a self-contained "black box" is created.
- When code and data are linked together in this fashion, an *object is* created. In other words, an object is the device that supports encapsulation.
- **In OOP** paradigm, **"object"** refers to a particular instance of a class.

Encapsulation

- Within an object, code, data, or both may be *private to that object* *or* *public*.
- *Private* code or data is known to and accessible only by another part of the object.
- When code or data is public, other parts of your program may access it even though it is defined within an object.

Polymorphism

- Polymorphism means to process objects differently based on their data type.
- In other words it means, one method with multiple implementation, for a certain class of action. And which implementation to be used is decided at runtime depending upon the situation (i.e., data type of the object).

- A real-world example of polymorphism is a thermostat. No matter what type of furnace your house has (gas, oil, electric, etc.), the thermostat works the same way. In this case, the thermostat (which is the interface) is the same no matter what type of furnace (method) you have.

Inheritance

- *Inheritance is the process by which one object can acquire the properties of another* object.

- It supports the concept of hierarchical classification.

- For example, a Red Delicious apple is part of the classification *apple, which in turn is part* of the *fruit class, which is under the larger class food.*

- Without the use of classifications, each object would have to define explicitly all of its characteristics.

This is a simple java program
Call this file “Example. Java”

```java
class Example{
    public static void main(String args[]){
     System.out.println("Hello Java");
    }
}
```

- For most computer languages, the name of the file that holds the source code to a program is immaterial. However, this is not the case with Java.
- The first thing that you must learn about Java is that the name you give to a source file is very important.
- For this example, the name of the source file should be **Example.java.**
- In Java, a source file is officially called a *compilation unit.*
- The Java compiler requires that a source file use the **.java filename extension.**

- Here, the name of the class defined by the program is also **Example.**

- **This is not a coincidence. In Java, all code must reside inside** a class. By convention, the name of the main class should match the name of the file that holds the program.

- You should also make sure that the capitalization of the filename matches the class name. The reason for this is that Java is case-sensitive.

- Explanation:

  public static void main(String args[])

- This line begins the **main( ) method.**
- All Java applications begin execution by calling **main( ).**
- The **public keyword is an *access modifier, which allows the programmer to control the*** visibility of class members. When a class member is preceded by **public, then that** member may be accessed by code outside the class in which it is declared.
- In this case, **main( ) must be declared as public, since it must be called by** code outside of its class when the program is started.

- **static** is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method. The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.

- The keyword **void simply tells the compiler that main() does not return a value.**

- **The main() is the method called when a Java application begins.**

- Any information that you need to pass to a method is received by variables specified within the set of parentheses that follow the name of the method. These variables are called *parameters.*
- In **main( ), there is only one parameter, a** complicated one.
- **String args[ ] declares a parameter named args, which is an array of** instances of the class **String.**
- *Arrays are collections of similar objects. Objects of type* **String store character strings. In this case, args receives any command-line arguments** present when the program is executed.

- One other point: **main( ) is simply a starting place for your program. A complex** program will have dozens of classes, only one of which will need to have a **main( )** method to get things started.

```
System.out.println("Hello Java");
```

- This line outputs the string “This is a simple Java program.” followed by a new line on the screen. Output is actually accomplished by the built-in **println() method. In this** case, **println( ) displays the string which is passed to it.**
- The line begins with **System.out.**
- **System is a predefined** class that provides access to the system, and **out is the output stream that is connected** to the console.

Basic Syntax

- About Java programs, it is very important to keep in mind the following points.
- **Case Sensitivity -** Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.
- **Class Names -** For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.
  Example: *class MyFirstJavaClass*
- **Method Names -** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.
  Example: *public void myMethodName()*

- **Program File Name -** Name of the program file should exactly match the class name.

  When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match your program will not compile).

  Example: Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as*'MyFirstJavaProgram.java'*
- **public static void main(String args[]) -** Java program processing starts from the main() method which is a mandatory part of every Java program.

Example2.java.

```
/*
   Here is another short example.
   Call this file "Example2.java".
*/

class Example2 {
  public static void main(String args []) {
    int num; // this declares a variable called num

    num = 100; // this assigns num the value 100

    System.out.println("This is num: " + num);

    num = num * 2;

    System.out.print("The value of num * 2 is ");

    System.out.println(num);
  }
}
```

```
This is num: 100
The value of num * 2 is 200
```

- Explanation:
- Using the + operator, you can join together as many items as you want within a single **println( ) statement.**

Java Identifiers

- All Java components require names. Names used for classes, variables and methods are called identifiers.
- In Java, there are several points to remember about identifiers. They are as follows:
- All identifiers should begin with a letter (A to Z or a to z), currency character ($) or an underscore (_).
- After the first character identifiers can have any combination of characters.

- A key word cannot be used as an identifier.
- Most importantly identifiers are case sensitive.
- Examples of legal identifiers: age, $salary, _value, __1_value
- Examples of illegal identifiers: 123abc, -salary

Java Keywords

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

Java Variables

- A variable is a container which holds the value while the java program is executed. A variable is assigned with a data type.

- Variable is a name of memory location.

- There are three types of variables in java:
  - local,
  - instance and
  - static.

Java Variables

1) **Local Variable**

- A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
- A local variable cannot be defined with "static" keyword.

2) Instance Variable

- A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.
- It is called instance variable because its value is instance specific and is not shared among instances.

3) Static variable

- A variable which is declared as static is called static variable. It cannot be local. You can create a single copy of static variable and share among all the instances of the class. Memory allocation for static variable happens only once when the class is loaded in the memory.

```java
class A{
int data=50;//instance variable
static int m=100;//static variable
void method(){
int n=90;//local variable
}
}//end of class
```

```java
class Simple{
public static void main(String[] args){
int a=10;
int b=10;
int c=a+b;
System.out.println(c);
}}
```

```
class Simple{
public static void main(String[] args){
int a=10;
float f=a;
System.out.println(a);
System.out.println(f);
}}
```

```
Output: 10 10.0
```

Type casting

```
class Simple{
public static void main(String[ ] args){
float f=10.5;
//int a=f;//Compile time error
int a=(int)f;
System.out.println(f);
System.out.println(a);
}}
```

```
Output: 10.5
        10
```

Thank you