Introduction to OOP

MODULE NAME: OBJECT ORIENTED PROGRAMMING WITH JAVA

MODULE CODE: CET 05102

INSTRUCTORS: JAMILA L.

Introduction to OOP

- OOP is a software development methodology in which a program is conceptualized as a group of objects that work together.
- OOP is one of the two major programming paradigms, the other is procedural (structured) programming.
- ***A programming paradigm is a fundamental style of computer programming.

Introduction cont…

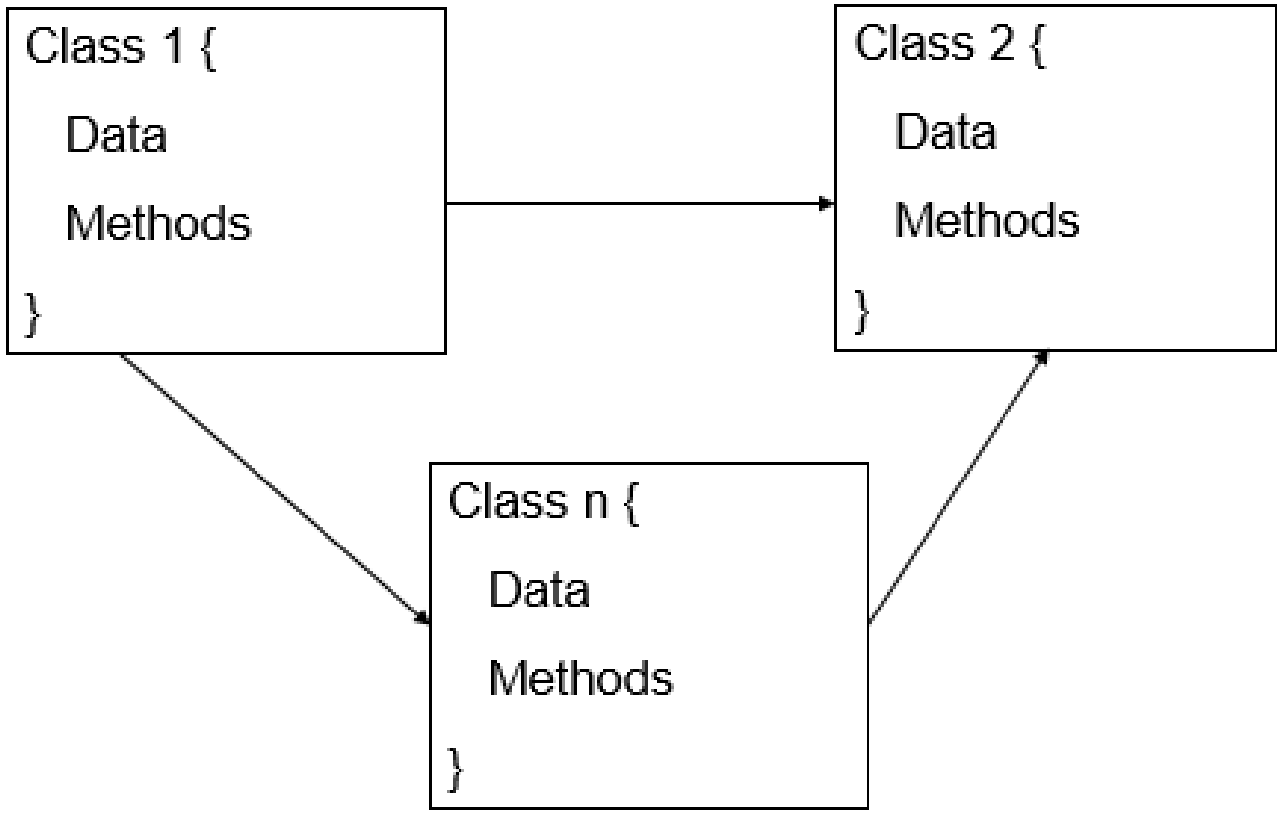
- The OOP methodology helps to organize complex programs through the use of inheritance, encapsulation and polymorphism.
- Objects are created using templates called classes, and they contain data and the statements required to use that data.
- OOP uses abstraction (in the form of classes and objects) to create models based on the real world environment.
- Objects are capable of passing messages, receiving messages, and processing data.

Introduction cont…

Therefore; OOP is defined in terms of objects (nouns) and the relationships between them.

- OOP combines both the data to be manipulated and the methods that manipulate them into objects, which are always dealt with together.
- Examples of OOP languages are
  - C++, C#, Ruby, Java, etc.

Structure of OOP Java program



OOP concepts

1. Objects
2. Classes
3. Inheritance
4. Encapsulation
5. Polymorphism
6. Abstraction

*** Encapsulation, inheritance and polymorphism are known as three principles of OOP.

Objects and classes

An **object** in software is a bundle of variables (states) and related methods (operations).

- OOP is modeled on the observation that in the physical world, objects are made up of many kinds of smaller objects.
- Real world objects are things that have
  1. State
  2. Behavior.

  E.g. Your mobile phone
  - State – make, model, colour
  - Behaviour – receiving, sending

Class

A class is a blueprint that defines the variables and methods common to all objects of a certain kind.

- E.g. ‘Your Mobile Phone’ is one object belonging to the class Phone.
- An object holds values for the variables defined in the class (e.g. make = “Nokia”).
- E.g. Car is the class; and wheels, speed limits, and mileage are their properties.

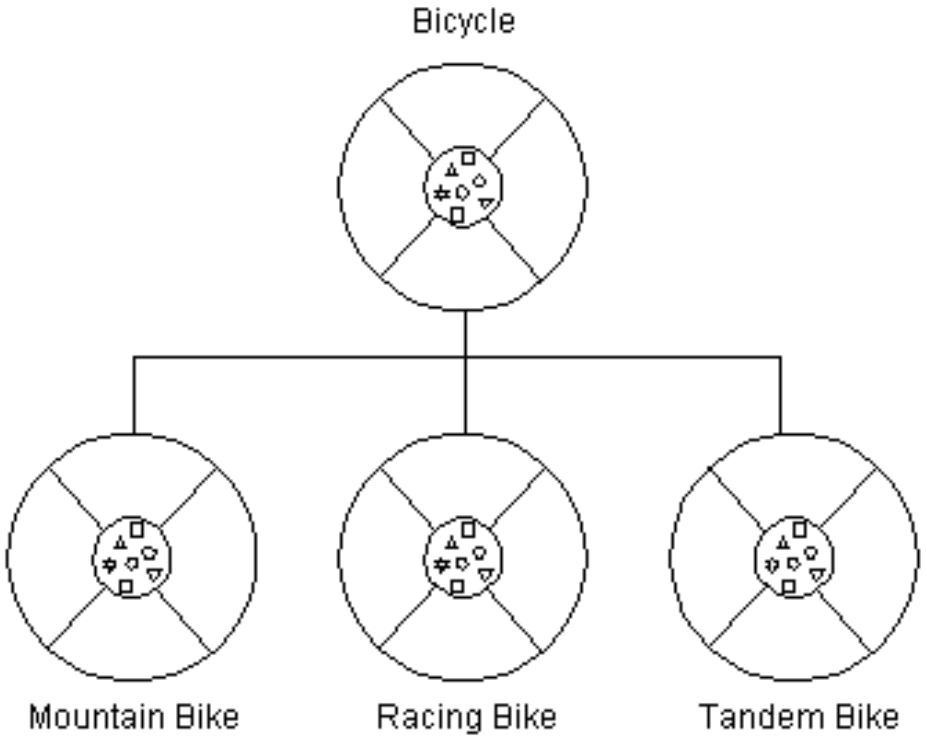
** An object is called an instance of the class.

Inheritance

Inheritance is a mechanism that enables one class to inherit all the behaviour and attributes of another class.

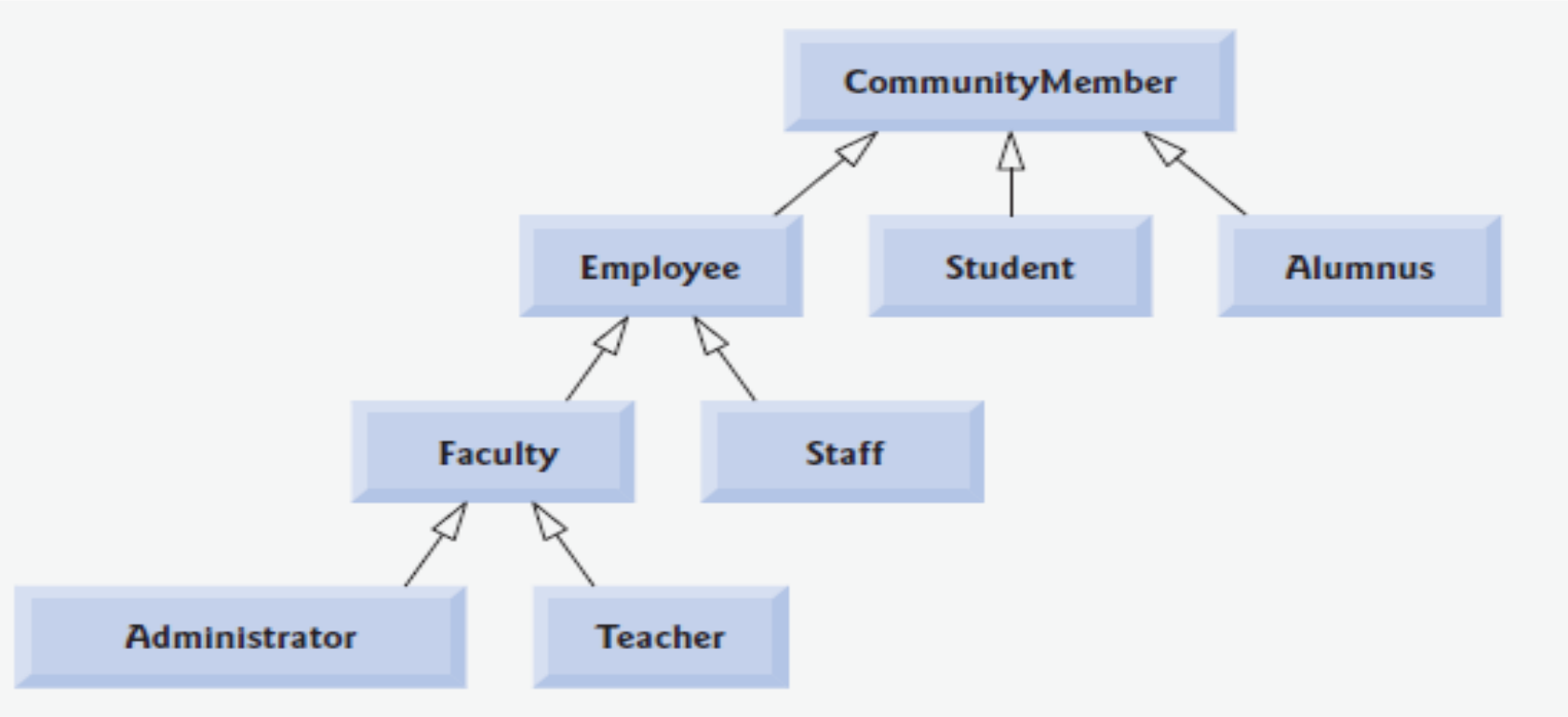
- Through inheritance, a class immediately has all the functionality of an existing class.
- Because of this, you must define only how the new class is different from an existing class.
- A class (subclass) can inherit attributes and methods from another class (superclass). Subclasses provide specialized behavior
- This mechanism Provides code reuse and Avoidance of duplication of data

Inheritance cont…



Inheritance hierarchy for Bicycle

Inheritance cont…



Inheritance hierarchy for university Community Members

Encapsulation

Encapsulation is the process that prevents class variables from being read or modified by other classes.

- The only way to use these variables is by calling methods of the class, if they are available.
- Encapsulation is made possible by access control modifiers, i.e.
  i. public
  ii. protected
  iii. default (no modifier) and
  iv. private

Polymorphism

Polymorphism is the ability to take many forms

- Allows you to define more than one method with the same name but different signatures in the same class.
- E.g. superclass ‘Parent’ has a method, printName().
- printName(String name) is also in subclass 'Child'.

Polymorphism in Java is mainly of 2 types:

- Overloading (Compile time polymorphism): the derived class provides the specific implementation of the method that is already provided by the base class or parent class.
- Overriding (Run time polymorphism): more than one method shares the same method name with a different signature in the class.

Abstraction

Data Abstraction is the property by virtue of which only the essential details are displayed to the user.

Data Abstraction can also be defined as the process of identifying only the required characteristics of an object, ignoring the irrelevant details.

Example:

- *Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the car speed or applying brakes will stop the car, but he does not know how on pressing the accelerator, the speed is actually increasing. He does not know about the inner mechanism of the car or the implementation of the accelerators, brakes etc. in the car. This is what abstraction is.*

Other concepts

- Dynamic Binding: In Dynamic binding compiler doesn't decide the method to be called. Overriding is a perfect example of dynamic binding. In overriding both parent and child classes have the same method. Dynamic binding is also called Late binding.

- Message Passing: in terms of computers is communication between processes. It is a form of communication used in object-oriented programming as well as parallel programming. Message passing in Java is like sending an object i.e. message from one thread to another thread.

Introduction to Java

Java

Java is a high-level (object-oriented) programming language originally developed by Sun Microsystems and released in 1995.

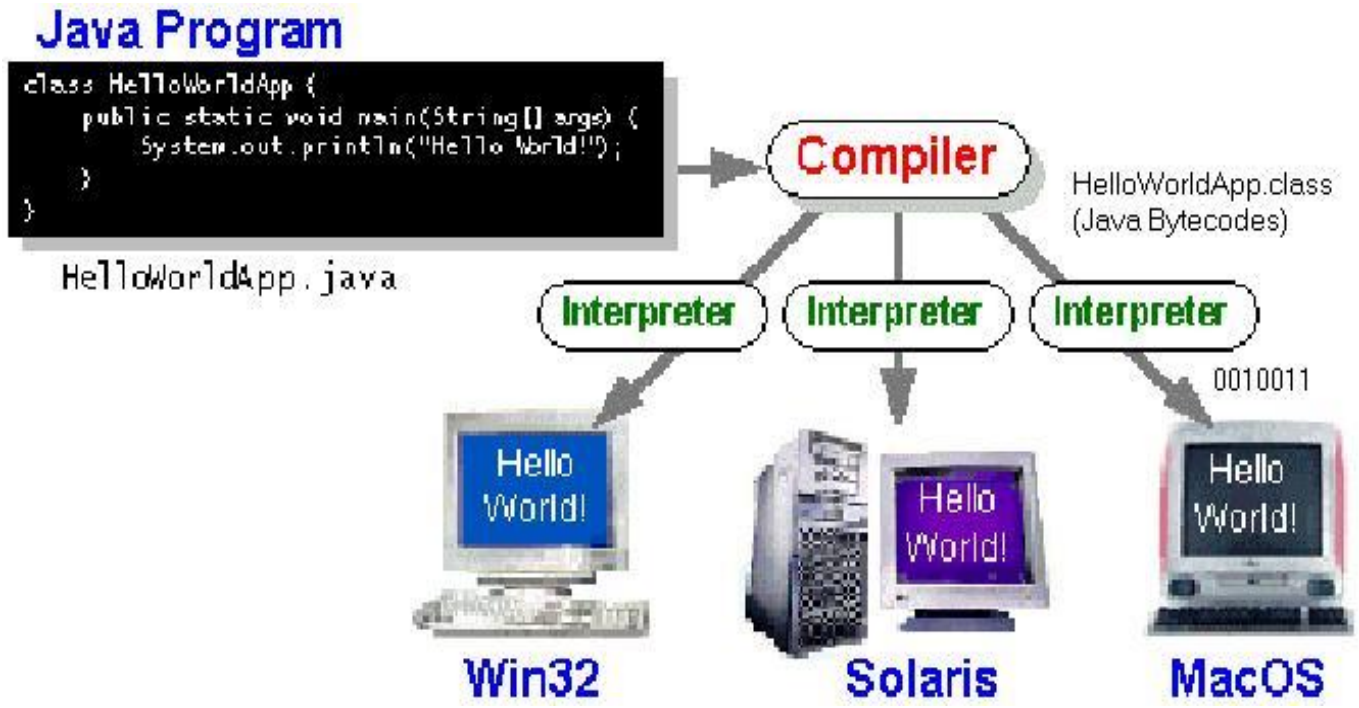
- Java is related to C++, which is direct descendant of C.
- From C, Java derives its syntax.
- Many of Java's object-oriented features were influenced by C++.

Java systems contain:

- Environment – hardware or software environment in which a program runs.
- Language – the Java language itself
- APIs – Java class libraries
- Java Virtual Machine (JVM)

Java cont...

- It is platform independent (Write Once Run Anywhere)



Advantages of Java

- Easy to learn, familiar C++ like syntax
- Pure object-oriented language
- Extensive documentation
- Faster application development because of code re-use
- Platform independence
- Automatic garbage collection

Simple Java application

```
//The program displays “Hello World! "

public class Hello {

 /*main method begins execution of Java application*/

 public static void main(String[] args) {

 // display “Hello World!"

 System.out.println(“Hello World!");

 }

}
```

Simple Java application cont...

Let us look at the statement that establishes the main() method.

- public – means that this method is available to other classes and objects
- static – means that main() is a class method
- void – means that main() method does not return a value
- main(String[] args) – a main method that takes one parameter, which is an array of strings.
  - This argument is used for program arguments.

Writing Java application

- The name of a source file is very important.
- In Java, all code must reside inside a class.
- By convention, the name of that class should match the name of the file that holds the program.
- However, it is name of the public class that must match the name of the Java source code file.
- A source code file uses the .java filename extension.
- Java is case-sensitive.

Compiling Java application

Java compiler first converts the source code into an intermediate code, known as bytecode or virtual machine code.

To run the bytecode, we need the Java Virtual Machine (JVM).

- JVM exists only inside the computer memory and runs on top of the Operating System. The bytecode is not machine specific.

The Java interpreter converts the bytecode into Machine code.

Assuming that you have typed your Java program in NetBeans IDE, follow the following steps:

- On menu bar, select Run → Compile File.
- To run a Java application, select Run → Run File.

Compiling Java application

The Java compiler breaks the line of code into text (words) is called **Java tokens**. These are the smallest element of the Java program.

For example, consider the following code.

```
public class Demo
{
public static void main(String args[])
{
System.out.println("javatpoint");
}
}
```

In the above code snippet, **public, class, Demo, {, static, void, main, (, String, args, [,],), System, ., out, println, javatpoint**, etc. are the Java tokens.

The Java compiler translates these tokens into Java bytecode. Further, these bytecodes are executed inside the interpreted Java environment.

Compiling Java application cont…

Alternatively, use the following steps:

- C:>\javac Hello.java (use terminal for Linux) .
- The javac compiler creates a file called Hello.class that contains the bytecode version of the program.
- To actually run program, you must use the Java interpreter, called java.

- C:>\java Hello
- The bytecode file name is written without filename extension
- When Java source code is compiled, each individual class is put into its own output file named after the class and using the .class extension.

Java keywords

Keywords: These are the **pre-defined** reserved words of any programming language.

- Each keyword has a special meaning. It is always written in lower case. Java provides the following keywords:

01. abstract	02. boolean	03. byte	04. break	05. class
06. case	07. catch	08. char	09. continue	10. default
11. do	12. double	13. else	14. extends	15. final
16. finally	17. float	18. for	19. if	20. implements
21. import	22. instanceof	23. int	24. interface	25. long
26. native	27. new	28. package	29. private	30. protected
31. public	32. return	33. short	34. static	35. super
36. switch	37. synchronized	38. this	39. thro	40. throws
41. transient	42. try	43. void	44. volatile	45. while
46. assert	47. const	48. enum	49. goto	50. strictfp

Comments in Java

- Comments make the code readable
- Give comments wherever necessary
- // – this is a single line (end-of-line) of comments
- /* – this is a block of comments (multiple-line) */

A closer look at the simple Java application

- All Java applications begin execution by calling main()
- A complex program will have dozens of classes, only one of which will need to have a main() method to get things started.
- Java interpreter runs only the class that has a main method
- println() is a built-in method used for handling output.
- System is a predefined class that provides access to the system, and
- out is the output stream that is connected to the console.

Main class and main method

- A class that contains main method is called main class.
- Each Java program must have one class that has a main method.
- The main method is the entry point, which is used to start the program by creating some objects and invoking their methods.

Syntax of Java class

- The syntax of a simple class is:
  - *modifier* class *ClassName* {
  - *modifier* data-type field1;
  - *modifier* data-type field2;
  - ...
  - *modifier* data-type fieldN;
  - *modifier* return-type methodName1(parameters){
  - //statements
  - }

Syntax of Java class cont…

modifier return-type methodName2(parameters){
//statements
}
…
modifier return-type methodN(parameters) {
//statements
}
}

Data types

- Data types in Java are divided into two categories - primitive types and reference types (sometimes called nonprimitive types).
- The primitive types are boolean, byte, char, short, int, long, float and double.
- All nonprimitive types are reference types, so classes, which specify the types of objects, are reference types.

Data types cont…

- Integers
  - byte (8 bit)
  - short (16 bit)
  - int (32 bit)
- Floating-point numbers
  - float (32 bit)
  - double (64 bit)
- Characters (char, 8 bit)
- Booleans (boolean, true or false)

Variable definition

AccessModifier DataType VariableName = Value;

Access modifier can be either public, protected, private, or default modifier.

- private int x = 5;
- char c = ‘x’;
- boolean b;
- final int I = 8; //constant variable

Java conventions

- Class names start with an uppercase
  - MyClass
- Variable names start with a lowercase
  - myVariable
- Method names start with a lowercase
  - myMethod
- Final variables are always capitalized (and initialized)
  - E.g., final float PI = 3.1415;
- Always give meaningful names to your variables, methods, and classes.
- Use indentation where necessary

Thank you!

for listen