B. Sc. (H) Computer Science Semester II
BHCS03 – Programming in JAVA

Topic:

- **Packages**
- **Interfaces**

Defining a Package

A Package is a collection of related classes. It helps organize your classes into a folder structure and make it easy to locate and use them. More importantly, it helps improve re-usability. Each package in Java has its unique name and organizes its classes and interfaces into a separate namespace, or name group.

Syntax:-

```
package nameOfPackage;
```

Package in java can be categorized in two form, built-in package and user-defined package.

There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Simple example of java package

The package keyword is used to create a package in java.

```
//save as Simple.java

package mypack;

public class Simple{

 public static void main(String args[]){

    System.out.println("Welcome to package");

   }

}
```

How to run java package program

You need to use fully qualified name e.g. mypack.Simple etc to run the class.

If you want to keep the package within the same directory, you can use . (dot).

To Compile: javac -d . Simple.java

To Run: java mypack.Simple

Output:Welcome to package

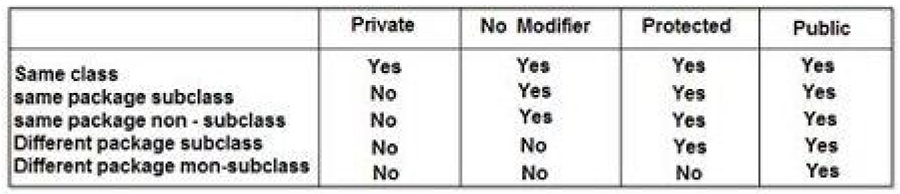
Access Protection

Access modifiers define the scope of the class and its members (data and methods). For example, private members are accessible within the same class members (methods). Java provides many levels of security that provides the visibility of members (variables and methods) within the classes, subclasses, and packages.

Packages

Packages are meant for encapsulating, it works as containers for classes and other subpackages. Class acts as containers for data and methods. There are four categories, provided by Java regarding the visibility of the class members between classes and packages:

1. Subclasses in the same package
2. Non-subclasses in the same package
3. Subclasses in different packages
4. Classes that are neither in the same package nor subclasses



	Private	No Modifier	Protected	Public
Same class	Yes	Yes	Yes	Yes
same package subclass	No	Yes	Yes	Yes
same package non - subclass	No	Yes	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package mon-subclass	No	No	No	Yes

Please solve/refer example discussed on pages 191-194 of reference book [*].

*Schildt, H. Java: The Complete Reference. 9th edition. McGraw-Hill Education.

Interface

An interface in Java is a blueprint of a class. It has static constants and abstract methods.The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body.

How to declare an interface?

An interface is declared by using the interface keyword. It provides total abstraction; means all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default. A class that implements an interface must implement all the methods declared in the interface.

Syntax:

```
interface <interface_name>{
   // declare constant fields      // declare methods that are abstract by default.

}
```

In this example, the Printable interface has only one method, and its implementation is provided in the A6 class.

```
interface printable{

void print();

}

class A6 implements printable{

public void print(){System.out.println("Hello");}


public static void main(String args[]){

A6 obj = new A6();

obj.print();

 }

}
```

Output:

```
Hello
```

Please solve/refer examples discussed on pages 197-200 of reference book [*].

*Schildt, H. . Java: The Complete Reference. 9th edition. McGraw-Hill Education.

Variables in Interfaces

All variables declared inside interface are implicitly public static final variables(constants). All methods declared inside Java Interfaces are implicitly public and abstract, even if you don't use public or abstract keyword. Interface can extend one or more other interface.

Please solve/refer examples discussed on pages 204-206 of reference book [*].

*Schildt, H. Java: The Complete Reference. 9th edition. McGraw-Hill Education.

Extending Interfaces

An interface can extend another interface in the same way that a class can extend another class. The extends keyword is used to extend an interface, and the child interface inherits the methods of the parent interface.

```
interface Printable{
void print();
}
interface Showable extends Printable{
void show();
}
class TestInterface4 implements Showable{
public void print(){System.out.println("Hello");}
public void show(){System.out.println("Welcome");}

public static void main(String args[]){
TestInterface4 obj = new TestInterface4();
obj.print();
obj.show();
 }
}
```

Output:

```
Hello
Welcome
```

Static Method in Interface

Since Java 8, we can have static method in interface. Let's see an example:

```
interface Drawable{

void draw();

static int cube(int x){return x*x*x;}
}

class Rectangle implements Drawable{

public void draw(){System.out.println("drawing rectangle");}

}



class TestInterfaceStatic{

public static void main(String args[]){

Drawable d=new Rectangle();

d.draw();

System.out.println(Drawable.cube(3));

}}
```

Output:

```
drawing rectangle
27
```

Nested Interface in Java

An interface can have another interface which is known as a nested interface. We will learn it in detail in the nested classes chapter.

```
interface printable{
 void print();
 interface MessagePrintable{
   void msg();
 }
}
```

References/Resources:

1. https://www.tutorialspoint.com
2. https://beginnersbook.com
3. https://docs.oracle.com

4. https://www.includehelp.com
5. Schildt, H. (2018). Java: The Complete Reference. 10th edition. McGraw-Hill Education.
6. Balaguruswamy, E. (2014). Programming with JAVA: A Primer. 5th edition. India: McGraw Hill Education
7. Horstmann, C. S. (2017). Core Java - Vol. I – Fundamentals (Vol. 10). Pearson Education

---

Assignment

Q1. What will be the output of the below program?

a)

```
interface A
{
   void myMethod();
}

class B
{
   public void myMethod()
   {
      System.out.println("My Method");
   }
}

class C extends B implements A
{

}

class MainClass
{
   public static void main(String[] args)
   {
      A a = new C();

      a.myMethod();
   }
}
```

b)

```
interface P
{
   String p = "PPPP";
```

```
    String methodP();
}

interface Q extends P
{
    String q = "QQQQ";

    String methodQ();
}

class R implements P, Q
{
    public String methodP()
    {
        return q+p;
    }

    public String methodQ()
    {
        return p+q;
    }
}

public class MainClass
{
    public static void main(String[] args)
    {
        R r = new R();

        System.out.println(r.methodP());

        System.out.println(r.methodQ());
    }
}
```

Q2. Create a class TwoDim which contains private members as x and y coordinates in package P1. Define the default constructor, a parameterized constructor and override toString() method to display the co-ordinates. Now reuse this class and in package P2 create another class ThreeDim, adding a new dimension as z as its private member. Define the constructors for the subclass and override toString() method in the subclass also. Write appropriate methods to show dynamic method dispatch. The main() function should be in a package P. (Try on machine also, if possible)

Q3. Define an abstract class Shape in package P1. Inherit two more classes: Rectangle in package P2 and Circle in package P3. Write a program to ask the user for the type of shape and then using the concept of dynamic method dispatch, display the area of the appropriate subclass. Also write appropriate methods to read the data. The main() function should not be in any package. (Try on machine also, if possible)

Q4. Define an interface shape which contains a function area(). Write the implementation of the interface for circle, rectangle and square. Also write the main() to test the interface. Can we declare variable in an Interface?

Q5. Can interfaces have constructors?