Mamoon Ali

Exercise 1

Encapsulation

Encapsulation is the process of wrapping code and data together as a single unit. It is a procedure of making fields in the class private and providing access to the fields through public methods. If the variable is declared private it cannot be accessed by anyone outside the class which means that it is hidden within the class. Encapsulation prevents code and data being accessed by other code defined outside the class.

E.g:

```
public class Test(){

private String name;

private String age;

public String getName(){

return name;

}

public int getAge(){

return age;

}

public void setName(String newName){

name = newName;

}

public void setAge(int newAge){

age = newAge;

}

}
```

The setters and getters (public set___ (), get___ ()) methods are the access points of the instance variables of the class so if any class that wants to access the variables it should be accessed through the setter and getter methods. If you want to set the name or age use the setter method. If you want to get the name or age use the getter method. E.g

```
public class RunTest{

public static void main(String[] args){
```

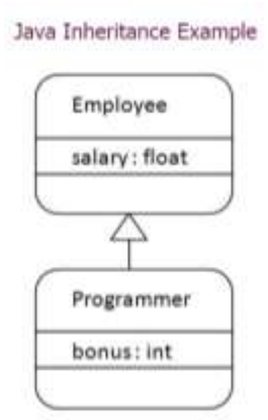
```
Test newTest = new Test();

newTest.setName("John");

newTest.setAge(22);

System.out.println("Name: " + newTest.getName() + " Age: " newTest.getAge());

}

}
```

Reference:
https://www.tutorialspoint.com/java/java_encapsulation.htm

Inheritance

Inheritance is a mechanism in which an object obtains all the properties and behaviours of parent object. The concept is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields.

Java Inheritance Example:



Programmer is the subclass and Employee is the superclass. Programmer is of type employee.

```
class Employee{
 float salary=40000;
}
class Programmer extends Employee{
 int bonus=10000;
 public static void main(String args[]){
   Programmer p=new Programmer();
   System.out.println("Programmer salary is:"+p.salary);
   System.out.println("Bonus of Programmer is:"+p.bonus);
}
}
```

```
Programmer salary is:40000.0
Bonus of programmer is:10000
```

Reference:
https://www.javatpoint.com/inheritance-in-java

Polymorphism

Polymorphism is the ability to present the same interface for differing underlying forms. The most common use of polymorphism occurs when a parent class reference is used to refer to a child class object. Any Java object that can pass more than one IS-A (relationship between two classes) test is considered to be polymorphic.

Example:

```
public interface Vegetarian{}

public class Animal{}

public class Deer extends Animal implements Vegetarian{}
```

The Deer class is considered polymorphic since it has multiple inheritance. It has the following relationships

A Deer IS-A Animal
A Deer IS-A Vegetarian
A Deer IS-A Deer
A Deer IS-A Object

All the reference variables d, a, v, o refer to the same Deer object in the heap.

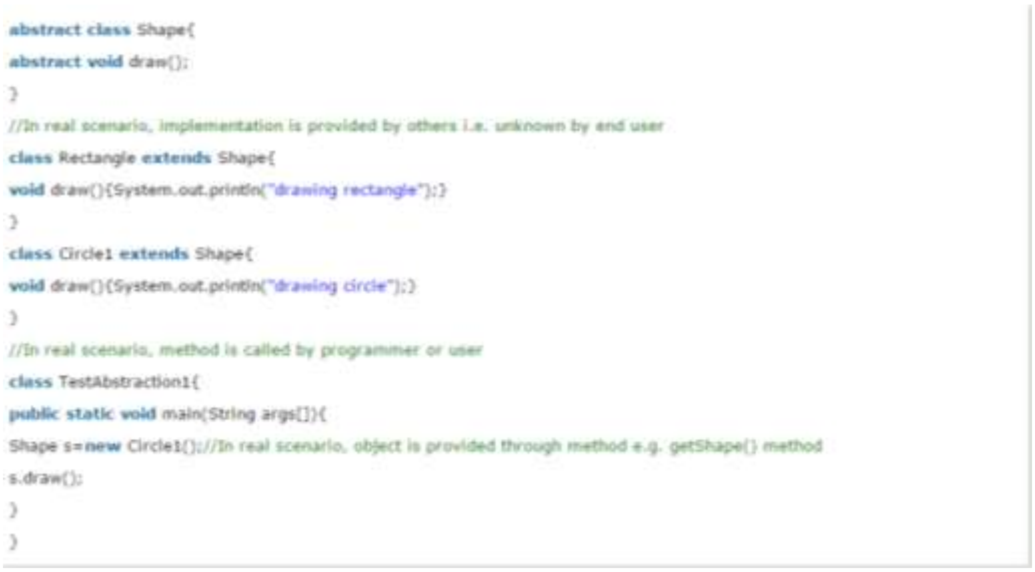
Reference:
https://www.tutorialspoint.com/java/java_polymorphism.htm

Abstraction

Abstraction is a process of hiding the implementation details from the user so only the functionality will be provided to the user. The user will have information on what the object does instead of how it does it. For example sending sms, you just type the text and send the message. You don't know the internal processing about the message delivery.

Mamoon Ali

In this example, Shape is the abstract class, its implementation is provided by the Rectangle and Circle classes. Mostly, we don't know about the implementation class



```
abstract class Shape{
abstract void draw();
}
//In real scenario, implementation is provided by others i.e. unknown by end user
class Rectangle extends Shape{
void draw(){System.out.println("drawing rectangle");}
}
class Circle1 extends Shape{
void draw(){System.out.println("drawing circle");}
}
//In real scenario, method is called by programmer or user
class TestAbstraction1{
public static void main(String args[]){
Shape s=new Circle1();//In real scenario, object is provided through method e.g. getShape() method
s.draw();
}
}
```

Print drawing circle

Reference:
https://www.javatpoint.com/abstract-class-in-java