Assignment 3

1. Explain polymorphism.

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

2. What is overloading?

Overloading occurs when two or more methods in one class have the same method name but different parameters.

3. What is overriding?

Overriding is an object-oriented programming feature that enables a child class to provide different implementation for a method that is already defined and/or implemented in its parent class or one of its parent classes.

4. What does the final mean in this method: public void doSomething(final Car aCar){}

Java always makes a copy of parameters before sending them to methods. This means the final doesn't mean any difference for the calling code. This only means that inside the method the variables “aCar” can not be reassigned.

5. Suppose in question 4, the Car class has a method setColor(Color color){…}, inside doSomething method, Can we call aCar.setColor(red);?

Yes

6. Can we declare a static variable inside a method?

No

7. What is the difference between interface and abstract class?

An abstract class can have both abstract method and not-abstract method, but interface just can have abstract class. A class can extend only one abstract class while a class can implement multiple interfaces. And abstract class can implement interfact while interface just can extend interface.

8. Can an abstract class be defined without any abstract methods?

No

9. Since there is no way to create an object of abstract class, what’s the point of constructors of abstract class?

If you define your own constructor with arguments inside an abstract class but forget to call your own constructor inside its derived class constructor then JVM will call the constructor by default.

So if you define your single or multi-argument constructor inside the abstract class then make sure to call the constructor inside the derived class constructor with the super keyword.

10. What is a native method?

A native method in Java is used to merge the efficiency and functions of C and C++ in the Java program. When the Java compiler did not implement or support some function, then, in that case, to increase the performance of the Java application, the native methods are used.

11. What is marker interface?

A marker interface is an interface that has no methods or constants inside it.

12. Why to override equals and hashCode methods?

We must override hashCode() in every class that overrides equals(). Failure to do so will result in a violation of the general contract for Object.hashCode(), which will prevent your class from functioning properly in conjunction with all hash-based collections, including HashMap, HashSet, and Hashtable.

13. What's the difference beween int and Integer?

int, being a primitive data type has got less flexibility. We can only store the binary value of an integer in it. Since Integer is a wrapper class for int data type, it gives us more flexibility in storing, converting and manipulating an int data. Integer is a class and thus it can call various in-built methods defined in the class. Variables of type Integer store references to Integer objects, just as with any other reference (object) type.

14. What is serialization?

Serialization is the process of converting an object into a stream of bytes to store the object or transmit it to memory, a database, or a file. Its main purpose is to save the state of an object in order to be able to recreate it when needed.

15. Create List and Map. List A contains 1,2,3,4,10(integer) . Map B contains ("a","1") ("b","2") ("c","10")   (key = string, value = string)
    Question: get a list which contains all the elements in list A, but not in map B.

```
public static List take(List<Integer> list,Map<String,String> map) {

            List newone = new ArrayList<Integer>();

            for(Integer tem: list) {
                newone.add(tem);
            }

            for(Map.Entry<String,String> set : map.entrySet()) {
                String rep =set.getValue();
                Integer num =Integer.parseInt(rep);
                if(list.contains(num)) {
                    newone.remove(num);
                }
            }
            return newone;
    }
```

16. Implement a group of classes that have common behavior/state as Shape. Create Circle, Rectangle and Square for now as later on we may need more shapes. They should have the ability to calculate the area. They should be able to compare using area. Please write a program to demonstrate the classes and comparison. You can use either abstract or interface. Comparator or Comparable interface.

```
interface Shape {
   abstract void compare(Shape s);
   abstract double area();
}
```

```
//your code goes here
public class Square implements Shape {
            double width;
            double area;

    public Square(double w){
        this.width = w;
    }
    public double area(){
        this.area = width*width;
        System.out.println(this.area);
        return this.area;
    }

            @Override
            public void compare(Shape s) {
                        // TODO Auto-generated method stub
                        if(this.area>s.area()) {
                                    System.out.println("Larger");
                        }else if(this.area==s.area()) {
                                    System.out.print("Equal");
                        }else {
                                    System.out.print("Smaller");
                        }
            }

}

class Circle implements Shape {
            double width;
            double area;
    public Circle(double w){
        this.width = w;
    }
    public double area() {
        this.area = Math.PI*this.width*this.width;
        System.out.println(this.area);
        return this.area;
    }
    @Override
            public void compare(Shape s) {
                        // TODO Auto-generated method stub
                        if(this.area>s.area()) {
                                    System.out.println("Larger");
                        }else if(this.area==s.area()) {
                                    System.out.print("Equal");
                        }else {
                                    System.out.print("Smaller");
                        }
            }
}

    public class Rectangle implements Shape {
                        double width;
```

```
		double length;
		double area;

	public Rectangle(double w,double l){
	    this.width = w;
	    this.length =l;
	}
	public double area(){
	    this.area = this.width*this.length;
	    System.out.println(this.area);
	    return this.area;
	}

		@Override
		public void compare(Shape s) {
			// TODO Auto-generated method stub
			if(this.area>s.area()) {
				System.out.println("Larger");
			}else if(this.area==s.area()) {
				System.out.print("Equal");
			}else {
				System.out.print("Smaller");
			}
		}
}
```