**Assignment 3**

1. Explain polymorphism.

Polymerphism is the ability of an object to take on many forms. For example, when a parent class reference is used to refer to a child class object. also when curtain task can be done in many different way, or in another word overloading.

1. What is overloading?

Overloading is when two method have the same name but different parameter.

1. What is overriding?

Overriding is when a subclass have the same method as declared in parent class.

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

This final prevents changing the reference to another object.

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

Yes.

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

No.

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

An interface allow programmer to define functionality but not implementing it. And abstract class allows programmer to create functionality that subclasses can implement or override. Abstract class can be inherited, interface can only be implemented.

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

Yes.

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

The subclasses of the abstract class can use this constructor. And abstract class also have non-abstract methods and instance variable, therefore a constructor is needed.

1. What is a native method?

Native ethods are methods that start in a language other than java.

1. What is marker interface?

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

1. Why to override equals and hashCode methods?

HashMap and HasSet use the hascode value of an object to find out how the object would be stored in the collection, and subsequently hascode is used to help locate the object in the collection. hashCode is used to find out the right bucket, and searching bucket for the right element is using equals(). Without overriding both method, the class will not function properly in conjunction with all hash-based collection like hashmap, hashset and hashtable.

1. What’s the difference beween int and Integer?

Int is a primitive type and Integer is the wrapper class of int type.

1. What is serialization?

Serialization in java allows us to convert an object to stream.

1. 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** **void** main(String[] args)

{

List<Integer> A = **new** ArrayList<Integer>();

A.add(1);

A.add(2);

A.add(3);

A.add(4);

A.add(10);

Map<String, Integer> B = **new** HashMap<String, Integer>();

B.put("a", 1);

B.put("b", 2);

B.put("c", 10);

List<Integer> C = **new** ArrayList<Integer>();

**for** (Integer i : A)

{

**if**(B.containsValue(i))

**continue**;

C.add(i);

}

}

1. 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.

**abstract** **public** **class** Shape **implements** Area, Comparable<Shape>

{

**double** area;

**public** **double** getArea()

{

**return** area;

}

**public** **int** compareTo(Shape shape2)

{

**if**(**this**.area == shape2.area)

**return** 0;

**else** **if** (**this**.area>shape2.area)

**return** -1;

**else** **return** 1;

}

}

**interface** Area

{

**public** **double** getArea();

}

**class** Circle **extends** Shape

{

**int** diameter;

**int** r;

**public** Circle(**int** diameter)

{

**this**.diameter = diameter;

r = diameter/2;

area = r\*r\*Math.***PI***;

}

}

**class** Rectangle **extends** Shape

{

**int** length1;

**int** length2;

**public** Rectangle(**int** length1, **int** length2)

{

**this**.length1 = length1;

**this**.length2 = length2;

area = length1\*length2;

}

}

**class** Square **extends** Shape

{

**int** length;

**public** Square(**int** length)

{

**this**.length = length;

area = length\*length;

}

}