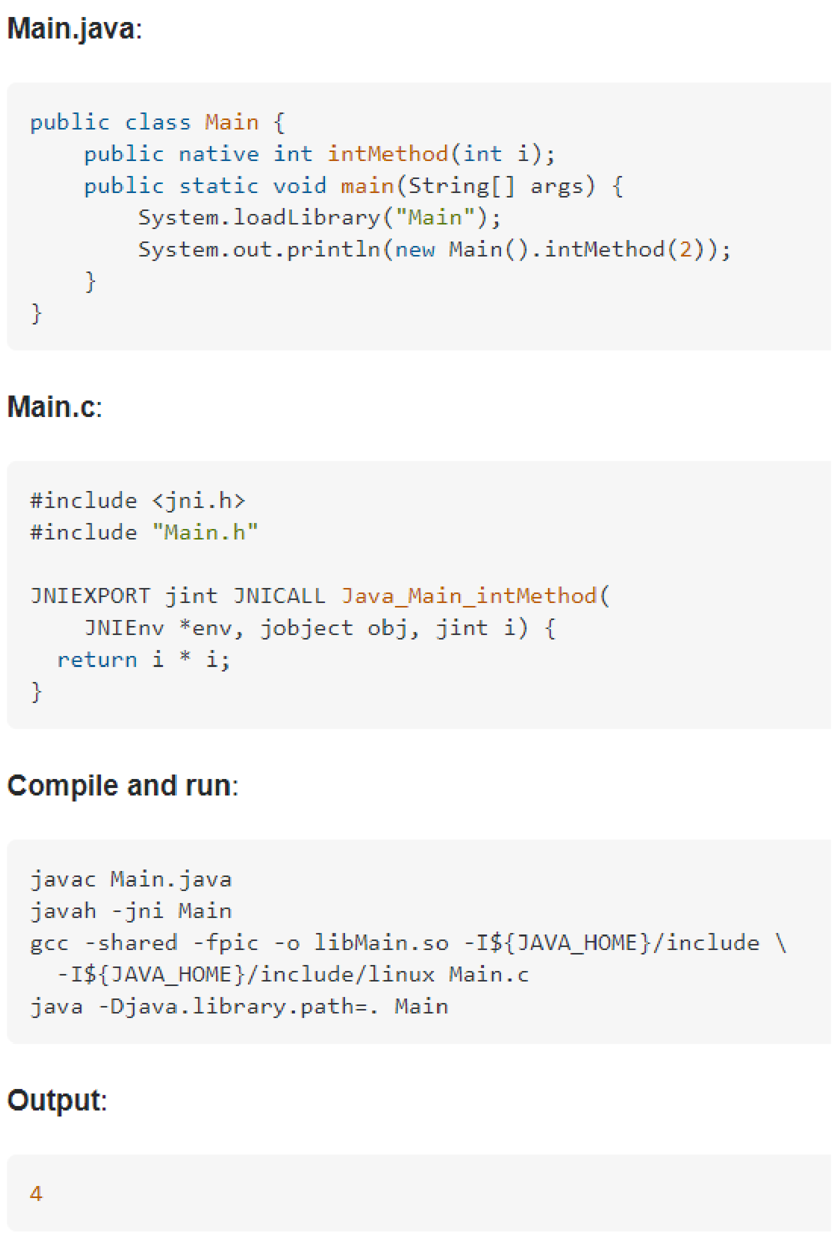
Assignment 3

1. Explain polymorphism.
   Polymorphism means "many forms". We can perform polymorphism in java by method overloading and method overriding.
2. What is overloading?
   If a class has multiple methods having the same name but different in parameters, it is known as Method Overloading.
3. What is overriding?
   If the subclass (child class) has the same method as declared in the parent class, it is known as method overriding in Java.
4. What does the final mean in this method: public void doSomething(**final** Car aCar){}
   aCar will be constant. The final keyword on a method parameter means absolutely nothing to the caller.
5. Suppose in question 4, the Car class has a method setColor(Color color){…}, inside doSomething method, Can we call aCar.setColor(red);?
   No, aCar can't be changed
6. Can we declare a static variable inside a method?
   You can't declare a static variable inside a method, static means that it's a variable/method of a class, it belongs to the whole class but not to one of its certain objects. This means that static keywords can be used only in a 'class scope' i.e. it doesn't have any sense inside methods. (c++ can)
7. What is the difference between an interface and an abstract class?
   The main difference is methods of a Java interface are implicitly abstract and cannot have implementations. A Java abstract class can have instance methods that implement a default behavior. Variables declared in a Java interface are by default final. An abstract class may contain non-final variables.
8. Can an abstract class be defined without any abstract methods?
   Yes, we can have an abstract class without Abstract Methods as both are independent concepts. Declaring a class abstract means that it can not be instantiated on its own and can only be subclassed.
9. Since there is no way to create an object of an abstract class, what's the point of constructors of abstract class?
   The main purpose of the constructor is to initialize the newly created object. In abstract class, we have an instance variable, abstract methods, and non-abstract methods. We need to initialize the non-abstract methods and instance variables, therefore abstract classes have a constructor.
10. What is a native method?
    A native method is a Java method whose implementation is also written in another programming language such as C/C++.



Main.java:

```java
public class Main {
    public native int intMethod(int i);
    public static void main(String[] args) {
        System.loadLibrary("Main");
        System.out.println(new Main().intMethod(2));
    }
}
```

Main.c:

```c
#include <jni.h>
#include "Main.h"

JNIEXPORT jint JNICALL Java_Main_intMethod(
    JNIEnv *env, jobject obj, jint i) {
  return i * i;
}
```

Compile and run:

```
javac Main.java
javah -jni Main
gcc -shared -fpic -o libMain.so -I${JAVA_HOME}/include \
  -I${JAVA_HOME}/include/linux Main.c
java -Djava.library.path=. Main
```

Output:

```
4
```

11. What is marker interface?
    A marker interface is an interface that has no methods or constants inside it. It provides run-time type information about objects, so the compiler and JVM have additional information about the object. A marker interface is also called a tagging interface.
    https://www.geeksforgeeks.org/marker-interface-java/
12. Why override equals and hashCode methods?
    If two objects are equal using the Object class equals method, then the hashcode method should give the same value for these two objects.
    If not, when we use the Map. A equal to B. But A and B can't find the same entity.
13. What's the difference between int and Integer?
    A Java both int and Integer are used to store integer type data the major difference between both is the type of int is primitive while the Integer is of class type.int helps in

storing integer value into memory. Integer helps in converting int into an object and to convert an object into int as per requirement. (Integer is used in all forms of Collections)

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. The reverse process is called deserialization.

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.

```java
package com.antra;
import java.util.*;


public class Test {
	public static void main(String[] args){
	  ArrayList<Integer> a = new ArrayList<Integer>(Arrays.asList(1,2,3,4,10));
	  Map< String,String> hm = new HashMap< String,String>();
	  hm.put("a", "1");
	  hm.put("b", "2");
	  hm.put("c", "10");

	  ArrayList<Integer> ans = new ArrayList<Integer>();
	  for(var i : a) {
		if(!hm.containsValue(i.toString())) {
			ans.add(i);
		}
	  }
	  System.out.println(ans);
	}
}
```

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 the area. Please write a program to demonstrate the classes and comparison. You can use either an abstract or interface. Comparator or Comparable interface.

https://www.geeksforgeeks.org/comparable-vs-comparator-in-java/

```java
package com.antra.shape;

import java.util.ArrayList;
import java.util.Collections;


abstract class Shape implements Comparable {
	int area;
	@Override
	public int compareTo(Object obj) {
		try {
			Shape shape = (Shape) obj;
			return this.area-shape.area;
		} catch (Exception e) {
			System.err.println(e);
		}
		return 0;
	}
}
class Circle extends Shape{
	public Circle(int radius) {
		this.area =(int) (radius*radius*3.14159);
	}
}
class Rectangle extends Shape{
	public Rectangle(int width,int height) {
		this.area =width*height;
	}
}
class Square extends Shape{
	public Square(int width) {
		this.area =width*width;
	}
}
public class Main {
	public static void main(String[] args){
		ArrayList arr = new ArrayList();

		arr.add(new Circle(2));
		arr.add(new Circle(3));
		arr.add(new Rectangle(5,4));
		arr.add(new Rectangle(6,3));
		arr.add(new Square(5));
		arr.add(new Square(4));
		Collections.sort(arr);
		for(Object i:arr) {
			Shape s = (Shape)i;
			System.out.print(s.getClass()+":");
			System.out.println(s.area);
		}
	}
}
```