1. Warm Up with Some Critical Concepts in Java Object-Oriented and Multithreading Programming: Choose the Correct Answer(s). Explain and justify:

a. Which of the following statements is correct for a method that overrides the following method:

```java
public void add(int a) {}
```

- The overriding method must return void
- The overriding method must return int
- The overriding method can return any type

b. Which of the following statements is correct for a method that overloads the following method:

```java
public void add(int a) {}
```

- The overloading method must return void
- The overloading method can return any type
- The overloading method must take int a as a parameter
- The overloading method can take any parameters

c. Given the following classes defined in separate files, what is the effect of compiling and running class Test?

```java
class Vehicle {
    public void drive() {
        System.out.println("Vehicle: drive");
    }
}
class Car extends Vehicle {
    public void drive() {
        System.out.println("Car: drive");
    }
}
public class Test {
    public static void main(String args []) {  
        Vehicle v;
        Car c;
        v = new Vehicle();
        c = new Car();
        v.drive();
        c.drive();
        v = c;
        v.drive();
    }
}
```

- Generates compile error at v = c
- Generates runtime error at v = c
- Prints: Vehicle : drive
  Car: drive
  Car: drive
- Prints: Vehicle: drive
  Car: drive
  Vehicle: drive
d. What is wrong with the following code?

- Since the method foo() does not catch the exception generated by the method baz(), it must declare the RuntimeException in a throws clause
- A try block cannot be followed by both a catch and a finally block
- An empty catch block is not allowed
- A catch block cannot follow a finally block
- A finally block must always follow one or more catch blocks

class MyException extends Exception {
    public class Qb4ab {
        public class bar() {
            try {
                baz();
            } finally {
                baz();
            } catch (MyException e) {
            }
        }
        public class baz() throws MyException {
            throw new MyException();
        }
        public class baz() throws RuntimeException {
            throw new RuntimeException();
        }
    }
}

```
abstract class MineBase {
    abstract void anmethod();
    static int t;
}

public class Mine extends MineBase {
    public static void main(String[] arg()) {
        int[] arr = new int[5];
        for(int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```

class A {}
class B extends A {}
class C extends A {}
public class Qba4 {
    public static void main(String[] args) {
        A x = new A();
        B y = new B();
        C z = new C();
        // (1) INSERT CODE HERE.
        }
    
}

class A {
    public A() {
        this();
    }
}
class B extends A {
    public boolean B(String msg) { return false; }
}
class C extends B {
    private C() { super(); }
    public C(String msg) { this();
    public C(int i) {
}
}

e. What is the result of compiling and running the following code:

- Prints a sequence of five zeros
- Generates runtime error since array arr has not been initialized
- Generates other error(s)

```java
abstract class MineBase {
    abstract void anmethod();
    static int t;
}

public class Mine extends MineBase {
    public static void main(String[] arg()) {
        int[] arr = new int[5];
        for(int i = 0; i < arr.length; i++) {
            System.out.println(arr[i]);
        }
    }
}
```

class A {}
class B extends A {}
class C extends A {}
public class Qba4 {
    public static void main(String[] args) {
        A x = new A();
        B y = new B();
        C z = new C();
        // (1) INSERT CODE HERE.
        }
    
}

class A {
    public A() {
        this();
    }
}
class B extends A {
    public boolean B(String msg) { return false; }
}
class C extends B {
    private C() { super(); }
    public C(String msg) { this();
    public C(int i) {
}
}

f. Which statement, when inserted at (1), will raise a runtime exception?

- x = y;
- z = x;
- y = (B) x;
- z = (C) y;
- y = (A) y;

class A {}
class B extends A {}
class C extends A {}
public class Qba4 {
    public static void main(String[] args) {
        A x = new A();
        B y = new B();
        C z = new C();
        // (1) INSERT CODE HERE.
        }
    
}

class A {
    public A() {
        this();
    }
}
class B extends A {
    public boolean B(String msg) { return false; }
}
class C extends B {
    private C() { super(); }
    public C(String msg) { this();
    public C(int i) {
}
}

g. Which statements are true about the following code?

- The code will fail to compile.
- The constructor in A that takes an int as an argument will never be called as a result of constructing an object of class B or C
- Class C defines three constructors
- Objects of class B cannot be constructed
- At most one of the constructors of each class is called as a result of constructing an object of class C

```java
class A {
    public A() {
        this();
    }
}
class B extends A {
    public boolean B(String msg) { return false; }
}
class C extends B {
    private C() { super(); }
    public C(String msg) { this();
    public C(int i) {
}
```
h. What is the output of the given function?

- Finally
- Exception
- Exception Finally
- No Output

```java
public void divide(int a, int b) {
    try {
        int c = a / b;
    } catch (Exception e) {
        System.out.print("Exception ");
    } finally {
        System.out.println("Finally");
    }
}
```

i. Which type constraints, when inserted at (1), will allow the class to compile?

- N extends Object
- N extends Comparable<N>
- N extends Object & Comparable<N>
- N extends Number
- N extends Number & Comparable<N>
- N extends Comparable<N> & Number
- N extends Integer
- N extends Integer & Comparable<N>

```java
class Interval<N> { // (1) INSERT TYPE CONSTRAINT HERE
    private N lower, upper;
    public void update(N value) {
        if (lower == null || value.compareTo(lower) < 0)
            lower = value;
        if (upper == null || value.compareTo(upper) > 0)
            upper = value;
    }
}
```

j. Given the following interface declaration, which declaration is valid?

```java
interface I {
    void setValue(int val);
    int getValue();
}
```

- ```java
class A extends I {
    int value;
    void setValue(int val) { value = val; }
    int getValue() { return value; }
}
```

- ```java
interface B extends I {
    void increment();
}
```

- ```java
abstract class C implements I {
    int getValue() { return 0; }
    abstract void increment();
}
```

- ```java
interface D implements I {
    void increment();
}
```
k. Which parameter declarations can be inserted at (1) so that the program compiles without warning?

```java
interface Wagger()
class Pet implements Wagger()
class Dog extends Pet()
class Cat extends Pet()

public class Q100_51 {
    public static void main(String[] args) {
        List<? super Pet> p = new ArrayList<Pet>();
        List<Dog> d = new ArrayList<Dog>();
        List<? extends Wagger> c = new ArrayList<Cat>();
        examine(p);
        examine(d);
        examine(c);
    }
    static void examine(__________ pets) { // (1)
        System.out.println("Your pets need urgent attention.");
    }
}
```

- List<? extends Pet>
- List<? super Pet>
- List<? extends Wagger>
- List<? super Wagger>
- List<?
- All of the above

l. Given the following code, which statements are true about the program?

```java
public class Person {
    protected transient String name;
    Person() { this.name = "NoName"; }
    Person(String name) { this.name = name; }
}

public class Student extends Person {
    protected long studNum;
    Student() {}
    Student(String name, long studNum) {
        super(name);
        this.studNum = studNum;
    }
}
```

```java
import java.io.Serializable;
public class GraduateStudent extends Student implements Serializable {
    private int year;
    GraduateStudent(String name, long studNum, int year) {
        super(name, studNum);
        this.year = year;
    }
    public String toString() {
        return "+ name + ", " + studNum + ", " + year + ");
    }
}
```
m. Which statements are true about the classes SupA, SubB, and SubC?

```java
class SupA<Z> {
    public List<? super Z> fuddle() { return null; }
    public List<X> scuddle(T t) { return null; }
}

class SubB<U> extends SupA<U> {
    public List<? super U> fuddle() { return null; }
    public List<X> scuddle(U t) { return null; }
}

class SubC<V> extends SupA<V> {
    public List<? super V> fuddle() { return null; }
    public List<X> extends Object scuddle(V t) { return null; }
}
```

• Class SubB will not compile
• Class SubC will not compile
• Class SubB will compile
• Class SubC will compile
• Class SubB overloads the methods in class SupA
• Class SubC overloads the methods in class SupA
• Class SubB overrides the methods in class SupA
• Class SubC overrides the methods in class SupA

n. Which interface is used to define a class that can execute within its own thread?

• Run
• Runnable
• Thread
• Threadable
• Executable

o. Which method is used to schedule a thread for execution?

• Init()
• Start()
• Run()

p. Which method(s) may cause a thread to stop execution?

• Sleep()
• Stop()
• Yield()
• Wait()
• Notify()

q. Given the following code, which of the following statements are true?

```java
public class Argg{
    public static void main(String argv[]){
        Argg a = new Argg();
        a.go();
    }
    public void go(){
        DSRoss ds1 = new DSRoss("one");
        ds1.start();
    }
}
class DSRoss extends Thread{
    private String sName="";
    DSRoss(String s){
        sName = s;
    }
    public void run(){
        wait();
        System.out.println("finished");
    }
    public void wait(){
        while(true){
            try{
                System.out.println("waiting");
                wait();
            }catch(InterruptedException ie){
                System.out.println(sName);
                notifyAll();
            }
        }
    }
}
```

• Fails to compile
• Prints “waiting”
• Prints “waiting” “finished”
• Compiles, but throws a run-time exception
2. **Practice with Inheritance and Polymorphism:**

Let’s use polymorphism to implement a very famous story line from our childhood – the Tortoise and Hare race. In this race, the Tortoise moves slowly and steadily with the motto of “slowly and steady wins the race”. On the other hand, the Hare spurs to the finish line with the philosophy of “jump, let the muscles pump”. Hence our two racers differ in the way they move.

To represent a contender, we define a base class *Racer* and two derived class *Tortoise* and *Hare*. The Racer class has three instance variables: String *ID* that identifies the type of racer (‘Tortoise’ and ‘Hare’), int *x* which denotes the x-axis position, and int *y* which denote the y-axis position. Inspect and study the activity’s archive, *HareTortoise.zip* and perform the following:

a. **Create a collection of racers:** before a race begins, the user enters, via a pre-programmed dialog box, the desired number of racers of each type. Your job is to create, instantiate, and add Tortoise and/or Hare objects to an ArrayList leveraging the concept of polymorphism. Complete the appropriate method to do so.

b. **Run the race:** run the race by moving and re-drawing all the racers towards the finish line. Complete the appropriate method to do so.

c. **Rank the racers:** at the end of the race, display the final positions of all the racers sorted by their proximity to the finish line (from the farthest to the closest). For this purpose, use the *sort* method of the Arrays class. Complete the appropriate method to do so.

3. **Practice with Socket Programming:**

In this exercise, you will code a group chat room. In this chat room, clients connect to a remote server with a designated IP and port number. The server accepts connections from an arbitrary number of clients. Any message sent from one client is broadcast to all other clients. You must leverage the concepts of socket programming and multi-threading to achieve this task. You may or may not create a GUI for your chat room.