**Exam 1**

**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Email: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

1. Analyze the following code:

**public class Test extends A {**

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

**Test t = new Test();**

**t.print();**

**}**

**}**

**class A {**

**String s;**

**A(String s) {**

**this.s = s;**

**}**

**public void print() {**

**System.out.println(s);**

**}**

**}**

* 1. The program does not compile because Test does not have a default constructor Test().
  2. The program has an implicit default constructor Test(), but it cannot be compiled, because its super class does not have a default constructor. The program would compile if the constructor in the class A were removed.
  3. The program would compile if a default constructor A(){ } is added to class A explicitly.
  4. The program compiles, but it has a runtime error due to the conflict on the method name print.

1. What is the output of running class C?

**class A {**

**public A() {**

**System.out.println(**

**"The default constructor of A is invoked");**

**}**

**}**

**class B extends A {**

**public B() {**

**A(); // "The default constructor of A is invoked"**

**System.out.println(**

**"The default constructor of B is invoked");**

**}**

**}**

**public class C {**

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

**B b = new B();**

**}**

**}**

1. Nothing displayed
2. "The default constructor of B is invoked"
3. "The default constructor of A is invoked""The default constructor of B is invoked"
4. "The default constructor of B is invoked""The default constructor of A is invoked"
5. "The default constructor of A is invoked"
6. Person p = new Person(); will throw a compilation error, because
7. Person is created as an abstract class //Can’t instantiate abstract class, only invoke its constructor on subclass.
8. Person is created as an interface
9. Person class only has a value constructor
10. Person class is a concrete class
11. The getValue() method is overridden in two ways. Which one is correct?

**I:**

**public class Test {**

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

**A a = new A();**

**System.out.println(a.getValue());**

**}**

**}**

**class B {**

**public String getValue() {**

**return "Any object";**

**}**

**}**

**class A extends B {**

**public Object getValue() {**

**return "A string";**

**}**

**}**

**II:**

**public class Test {**

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

**A a = new A();**

**System.out.println(a.getValue());**

**}**

**}**

**class B {**

**public Object getValue() {**

**return "Any object";**

**}**

**}**

**class A extends B {**

**public String getValue() {**

**return "A string";**

**}**

**}**

1. I
2. II
3. Both I and II
4. Neither
5. Analyze the following code:

**public class Test {**

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

**Object a1 = new A();**

**Object a2 = new Object();**

**System.out.println(a1);**

**System.out.println(a2);**

**}**

**}**

**class A {**

**int x;**

**public String toString() {**

**return "A's x is " + x;**

**}**

**}**

1. The program cannot be compiled, because System.out.println(a1) is wrong and it should be replaced by System.out.println(a1.toString());
2. When executing System.out.println(a1), the toString() method in the Object class is invoked.
3. When executing System.out.println(a2), the toString() method in the Object class is invoked.
4. When executing System.out.println(a1), the toString() method in the A class is invoked.
5. The \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ method is used to determine which object is the source object triggering the event when the button is clicked.
6. public int actionPerformed(ActionEvent e) { return 0;}
7. public void actionPerformed(ItemEvent e) { return; }
8. public String actionPerformed(ActionEvent e) { return “Hello”;}
9. public void actionPerformed(ActionEvent e) { return;} // “return;” on it’s own
10. (True/False) You can always pass an instance of a superclass to a parameter of its subclass type. This feature is known as polymorphism.
11. A class that is declared \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ cannot be instantiated.
12. Analyze and the select the correct answer/s:

**public class A extends B {**

**B();  
 }**

**class B {**

**public B(String s) {**

**}**

**}**

1. The program has a compilation error because A does not have a default constructor.

B. The program has a compilation error because the default constructor of A invokes the default constructor of B, but B does not have a default constructor.

C. The program would compile fine if you add the following constructor into A: A(String s) { }

D. The program would compile fine if you add the following constructor into A: A(String s) { super(s); }

1. What is the output of the following code:

**public class C extends B {**

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

**new C();**

**}**

**public C() {**

**System.out.println("C");**

**}**

**}**

**class B extends A {**

**public B() {**

**this("Java");**

**System.out.println("B");**

**}**

**public B(String s) {**

**System.out.println(s);**

**}**

**}**

**class A {**

**public A() {**

**System.out.println("A");**

**}**

**}**

1. What is the output of the following code

public class TestMystery {  
 public static void main(String [] args) {  
 double[] list = {20, 9, 14.9, 16.6, 50.7, -1.5, 3.7};  
 Mystery.mystery(list);  
 for (double i : list)  
 System.out.println(i);  
 }  
}  
class Mystery {  
 public static void mystery(double[] list) {  
 for (int i = 0; i < list.length - 1; i++) {  
 double currentMin = list[i];  
 int currentMinIndex = i;  
 for (int j = i + 1; j < list.length; j++) {  
 if (currentMin < list[j]) {  
 currentMin = list[j];  
 currentMinIndex = j;   
 }  
 }  
 if (currentMinIndex != i) {  
 list[currentMinIndex] = list[i];  
 list[i] = currentMin;  
 }  
 }  
 }  
 }

Output:

Q1. b, c

Q2.  c, additional option of "None of the above" is also acceptable

Q3.  a,b, c

Q4. b

Q5. c,d

Q6. d

Q7. false

Q8. abstract

Q9. b,d

Q10. A, Java B C

Q11.

50.7

20.0

16.6

14.9

9.0

3.7

-1.5