Section A

1.

```
class Conditional{
    public static void main(String[] args){
      int count1 = 10;
      int count2 = 20;
      int result;
      boolean condition = false;
      result = condition ? count1 : count2;
            System.out.println(result);
   }}
```

a. Compile Error
b. 10
c. 20
d. 10,20
e. false

2)

```
class A {
   public void m() {
      System.out.println("A m");
}}
interface B implements A{ //Line 1
   public void n();
   public void m(); //Line 2
}
class C implements B {
   public void n() {
      System.out.println("C n");
   }
}
class Test {
   public static void main(String[] args) {

      C c = new C();
      c.n();
      c.m(); //Line 3
   }}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. Compile Error at Line 3
d. C n , A m

3)

```
class Logical {
   public static void main(String args[]) {
      boolean a = true;
      boolean b = false;
      System.out.println("a && b = " + (a && b));
      System.out.println("a || b = " + (a || b));
      System.out.println("!(a && b) = " + !(a && b));
   }}
```

a. a && b = false
   a || b = true
   !(a && b) = true

b. a && b = false
   a || b = false
   !(a && b) = true

c. a && b = true
   a || b = true

!(a && b) = true

d. a && b = false
a || b = true
!(a && b) = false

e. Compile Error

4)

```
class ConditionalDemo {
   public static void conditional(boolean b) {
      boolean value = b;
      String a = (value) ? ("BBB") : ((value) ? ("CCC") : ("DDD"));
      System.out.println(a);
   }
   public static void main(String[] args) {
      conditional(false);
   }
}
```

a. Compile error
b. BBB
c. CCC
d. DDD
e. BBB,CCC

5)

```
class Switch {
   public static void main(String[] args) {
      int i = 3;
      switch (i) {
         case 1:
            System.out.println("i is equal to 1");
            break;
         case 2:
            System.out.println("i is equal to 2");
            break;
         case 3:
            System.out.println("i is equal to 3");
         case 4:
            System.out.println("i is equal to 4");
         default:
            System.out.println("Still no idea");
            break;
      }
   }
}
```

a. i is equal to 3
i is equal to 4
Still no idea

b. i is equal to 4
i is equal to 4
Still no idea

c. i is equal to 4
i is equal to 3
Still no idea

d. i is equal to 3
i is equal to 4

e. Compile Error

6)

```
interface X {
   public void m(); //Line 1
   public abstract void n();
```

```
}
interface Y {
   public abstract void q();
}
abstract class A implements X,Y{ //Line 2
   public void n(){
      System.out.println("A n");
   }
}
class B extends A{ //Line 3
   public void q(){
      System.out.println("B q");
   }
}
class Test {
   public static void main(String[] args) {
      B b = new B();
      b.q();
      b.n();
   }
}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. Compile Error at Line 3
d. B q , A n

7)

```
class IfCondition{
    public static void main(String[] args) {
        boolean condition;
        if (condition = false) {
            System.out.print("A");
        } else if (condition) {
            System.out.print("B");
        }  else if (!condition) {
            System.out.print("C");
        } else {
            System.out.print("D");
        }
    }
}
```

a. D
b. C
c. B
d. A
e. None of the above

8)

```
class A {
   public static void main(String[] args) {
      final int x = 10;
      switch (10) {
         case x:
            System.out.println("case 1"); break;
         case (x + 1):
            System.out.println("case 2");
      }
   }
}
```

a. Compile Error
b. RunTime Error
c. case 1
d. case 1,case 2
e. case 2

9)

```
class DataStructures {
   public static void main(String[] args) {
      int structure[][] = new int[4][];
      structure[0] = new int[1];
      structure[1] = new int[2];
      structure[2] = new int[3];
      structure[3] = new int[2];
      structure[0][0] = 10;
      structure[1][1] = 20;
      structure[2][1] = 3;
      structure[3][1] = 12;
      System.out.print(structure[0][0]);
      System.out.print(structure[1][1]);
      System.out.print(structure[2][1]);
      System.out.print(structure[3][1]);
      System.out.print(structure[2][2]);
   }
}
```

a. Compile Error
b. 10203120
c. 10320120
d. 10012320
e. ArrayIndexOutOfBoundsException

10) Given:

```
1. class SuperFoo {
2.   SuperFoo doStuff(int x) {
3.     return new SuperFoo();
4.   }
5. }
6.
7. class Foo extends SuperFoo {
8.   // insert code here
9. }
```

And four declarations:
I. Foo doStuff(int x) { return new Foo(); }
II. Foo doStuff(int x) { return new SuperFoo(); }
III. SuperFoo doStuff(int x) { return new Foo(); }
IV. SuperFoo doStuff(int y) { return new SuperFoo(); }

Which, inserted independently at line 8, will compile?

a. Only I.
b. Only IV.
c. Only I and III.
d. Only I, II, and III.
e. Only I, III, and IV. (*)
f. All four declarations will compile.

11) How can you force garbage collection of an object?

a. Garbage collection cannot be forced.
b. Call System.gc().
c. Call System.gc() passing in a reference to the object to be garbage collected.
d. Call Runtime.gc().
e. Set all references to the object to new values(null, for example).

12) In Java, an abstract class cannot be sub-classed.

a. True
b. False

13) What is the output of this code fragment?

```
int X=3; int Y =10;
System.out.println(y%x);
```

a. 0
b. 1
c. 2
d. 3

14) What is the output of following code fragment?

```
class Mammal{
void  makeNoise(long  x){
     System.out.print("Mammal Big Noice ,");
  }}
class Zebra extends Mammal{
  void  makeNoise(long x){
     System.out.print("Zebra Big Noice ,");
  }
  void  makeNoise(int x){
     System.out.print("Zebra Small Noice ,");
  }}
class Test{
   public static void main(String[] args) {
      Mammal a = new Mammal();
      Zebra b = new Zebra();
      Mammal c = new Zebra();
       byte noice = 10;
       a. makeNoise(noice);
       b. makeNoise(noice);
       c. makeNoise(noice);
}}
```

a. Mammal Big Noice ,Zebra Small Noice ,Zebra Big Noice ,
b. Mammal Big Noice ,Mammal Big Noice ,Zebra Big Noice ,
c. Mammal Big Noice ,Zebra Big Noice ,Zebra Small Noice ,
d. Compilation fails
e. Runtime Exception

15)

```
class Pow {}
class Wow extends Pow {}
class American {
   void bow(Pow p) {
      System.out.println("American");
   }}
```

```
class Redindian extends American {
   void bow(Wow w) {
      System.out.println("Redindian");
   }}
class Test {
   public static void main(String[] args) {
      American a = new American();
      Wow w = new Wow();
      a.bow(w);
   }}
```

a. American
b. Redindian
c. Compilation fails
d. Runtime Exception

16)

```
class Pow {}
class Wow extends Pow {}
class American {
   void bow(Pow p) {
      System.out.println("American");
   }
}
class Redindian extends American {
   void bow(Wow w) {
      System.out.println("Redindian");
   }
}
class Test {
   public static void main(String[] args) {
      Redindian a = new Redindian();
      Wow w = new Wow();
      a.bow(w);
   }
}
```

a. American
b. Redindian
c. Compilation fails
d. Runtime Exception

17)

```
class Plant {
        Plant() {
                System.out.println("Plant created");
        }
}
class Tree extends Plant {
        Tree() {
                System.out.println("Tree created");
            super();
        }
}
class Test {
        public static void main(String args[]) {
                Tree tree = new Tree();
        }
}
```

Please choose only one answer:

a. Plant created
b. Tree created
c. Tree created
d. Plant created
e. RuntimeException
f. Compilation fails

18)

```
class Base {
        private Base() {System.out.print("Base");}
}
 class Derived extends Base {
         Derived() {System.out.print("Derived");}
        public static void main(String[] args) {
                new Derived();
        }
}
```

a. Please choose only one answer:
b. BaseDerived
c. Derived
d. Exception is thrown at runtime
e. Compilation fails

19)

```
class Tester {
        static void test(int[] a) {
                int[] b = new int[2];
                a = b;
                System.out.print(b.length);
                System.out.print(a.length);
        }
        public static void main(String[] args) {
                int[] a = new int[5];
                test(a);
                System.out.print(a.length);
        }
}
```

Please choose only one answer:

a. 225
b. 255
c. 200

20)

```
class Creature {}
class Bird extends Creature {}
class Parrot extends Bird {}
class Foo {
   void chirp(Creature cr) {
      System.out.println("Foo Creature");
   }
   void chirp(Bird brd) {
      System.out.println("Foo Bird");
   }
}
class Boo extends Foo {
   void chirp(Parrot y) {
      System.out.println("Boo Parrot");
   }
}
class Test {
   public static void main(String[] args) {
```

```
        Foo f = new Boo();
        Parrot p = new Parrot();
        f.chirp(p);
    }
}
```

Please choose only one answer:

a. Foo Creature
b. Foo Bird
c. Boo Parrot
d. Compilation fails

21)

```
class X {}
    class Y extends X {}
    class Z extends Y {}
    class Vehicle{
      void m(X x){System.out.println("Vehicle X");}
      void m(Y y){System.out.println("Vehicle Y");}
    }
    class Car extends Vehicle{
      void m(Z z){System.out.println("Car  Z"); }}

    class Test{
       public static void main(String[] args) {
          Vehicle v = new Car();
          Y y = new Z();
            v.m(y);
       }
    }
```

Please choose only one answer:

a. Vehicle X
b. Vehicle Y
c. Car Z
d. Compilation fails

22)

```
class Animalia{
     void m(){
        System.out.println("Animalia m");
     }}
    class Aves extends Animalia{
     void m(){
        System.out.println("Aves m");
     }}
```

```
class Test {
   public static void main(String[] args) {
      Animalia p = new Animalia();
      Aves     q = new Aves();
      Animalia r = new Aves();
      p.m();
      q.m();
      r.m();
   }
}
```

Please choose only one answer:

a. Animalia m
Aves m
Aves m

b. Aves m
Animalia m
Aves m

c. Aves m
Aves m
Animalia m

d. Compilation fails

23)
```
package powerframe;

class X {
      void method(){
        System.out.println("X-method");
 }}
class Xor extends X{
  void method(){
    System.out.println("Xor-method");
 }}
class Test {
   public static void main(String[] args) {
      X a = new X();
      Xor b = new Xor();
      X a1= new Xor();
      a.method();
      b.method();
      a1.method();
   }}
```

a. X-method
Xor-method
Xor-method

b. X-method
X-method
Xor-method

c. X-method
Xor-method

X-method

d. Compilation fails

24)

```
class Super {
  static void m(){
     System.out.println("Super m");
  }}
class Sub extends Super{
  static void m(){
     System.out.println("Sub m");
  }}
class Test {
   public static void main(String[] args) {
      Super a = new Super();
      Sub b   = new Sub();
      Super c= new Sub();
      a.m();
      b.m();
      c.m();
   }}
```

.

a. Super m

Sub m

Sub m

b. Super m

Sub m

Super m

c. Super m

Super m

Super m

d. Compilation fails

25)

```
class Superhero{
        void laserAttack(){
     System.out.println("Superhero laserAttack");
  }}
class Superman extends Superhero{
   protected  void laserAttack(){
     System.out.println("Superman laserAttack");
  }
   public static void main(String[] args) {
      Superhero sh = new Superman();
            sh.laserAttack();
   }
}
```

a. Superhero laserAttack

b. Superman laserAttack

c. Compilation fails

d. Runtime Error

26)

```
class Weapon{
     void fire(){
     System.out.println("Weapon fire");}
}
class LaserWeapon extends Weapon{
```

```
    void fire(){
       System.out.println("LaserWeapon fire");}
}
class Superhero{
        Weapon laserAttack(){
    System.out.println("Superhero laserAttack");
        return new Weapon();
 }}
class Superman extends Superhero{
       LaserWeapon laserAttack(){
    System.out.println("Superman laserAttack");
        return new LaserWeapon();
 }
   public static void main(String[] args) {
      Superhero sh = new Superman();
          sh.laserAttack().fire();
   }
}
```

a. Compilation fails

b. Superhero laserAttack
Weapon fire

c. Superhero laserAttack
LaserWeapon fire

d. Superman laserAttack
Weapon fire

e. Superman laserAttack
LaserWeapon fire

27)

```
class Foo{
     static void m(){
       int x =67;
       return;
        System.out.println(x);
     }
      public static void main(String[] args) {
         System.out.println("main method-start");
         m();
         System.out.println("main method - over");
       }
}
```

a. main method-start
67
Runtime Exception

b. main method-start
67
main method- over

c. Compilation fails

d. main method-start
e. Runtime Exception

28)

```
import javax.swing.*;
package powerframe;
class A {
   public static void main(String[] args) {
     JFrame frame = new JFrame("Power Frame");
         frame.setVisible(true);}}
```

a. java JFrame will appear
b. No output
c. Compile Error
d. Runtime Error

29)

```
class DispData {
   void display(long longdata){
      System.out.println("print longdata ");}
   void display(int intdata){
      System.out.println("print intdata ");}
   public static void main(String[] args) {
  DispData disp = new DispData();
      byte bytedata = 70;
       disp.display(bytedata);
      System.out.println();}}
```

a. print intdata
b. print longdata
c. Compile Error
d. Runtime Error

30)

```
class Foo{
   static int x = 10;
       void changeVal2(){
            x=68;
            changeVal1();
       }
 static void changeVal1(){
          x=35;
 }
   public static void main(String[] args) {
        System.out.println(x);
         Foo f = new Foo();
           f.changeVal2();
        System.out.println(x);

   }
```

```
}
```

a. 10

64
35
35

b. 10

68

c. 10

35

d. Compilation fails

31)

```
class Sprite{
    int x = 10;
       void moveUp(){
           x=68;
           moveDown();
       }
 static void moveDown(){
          x=35;
 }
   public static void main(String[] args) {

        Sprite s = new Sprite();
          s.moveUp();
        System.out.println(s.x);

   }
}
```

a. 10

64
35
35

b. 10

68

c. 10

35

d. Compilation fails

32)

```
package powerframe;

class ValuePower{
   static int val;
  int getSet(int val){
     ValuePower.val = val;
     return val;
  }
   public static void main(String[] args) {
     ValuePower v = new ValuePower();
        System.out.println(v.getSet(40));
        System.out.println(v.getSet(12));
```

```
        System.out.println(v.getSet(25));
        System.out.println(v.val);

    }
}
```

a. 40
12
25
0

b. 12
40
25
25

c. 40
12
25
25

d. Compile Error

e. Run Time Error

33)

```
abstract class A {
   public abstract void m();
}
class B extends A {
   public void q() {
      System.out.println("B q");
   }
   public void m() {
      System.out.println("B m");
   }
}
abstract class C extends B { //Line 1
   public abstract void q(); //Line 2
}
class D extends C { //Line 3
   public void m() {
      System.out.println("D m");
   }
}
class Test {
   public static void main(String[] args) {
      D d = new D();
      d.m();
   }
}
```

a. Compile Error at Line 1

b. Compile Error at Line 2

c. Compile Error at Line 3

d. D m

34)

```
class SecurityValue{
   int secVal;
  int getSecVal(){
     return this.secVal;
  }
  void setSecVal(int secVal){
     if(secVal>200){
        System.out.println("Insecure value");
        return;
```

```
        }
        this.secVal=secVal;
    }
    public static void main(String[] args) {

        SecurityValue sv = new SecurityValue();
            System.out.println(sv.getSecVal());
                sv.setSecVal(300);
                sv.setSecVal(70);
                sv.setSecVal(400);
                sv.setSecVal(34);
            System.out.println(sv.getSecVal());
    }
}
```

a. 0

```
Insecure value
Insecure value
34
```

b. Compile Error
c. Runtime Error
d. 300

```
Insecure value
Insecure value
400
```

35)

```
class X {}
class Y extends X{
 void m(){
    System.out.print("code");
 }
}
class A{
  public static void main(String args[]){
     X x = new X();
     Y y = new Y();
     X z = new Y();
     x.m(); // line 1
     y.m(); // line 2
     x.m(); // line 3

  }
}
```

a. compile Error at line 1
b. compile Error at line 2
c. compile Error at line 3
d. Runtime Error at line 1
e. codecodecode

36)

```
class A {}
class X extends A{
   void voice (){
      System.out.println("Brr");
   }}
```

```
class Y extends A{
    void voice(){
       System.out.println("Brr");
    }}
class Insp{
    Y g;
    void impls(){
      g.voice();// line one
    }}
class Test {
   public static void main(String[] args) {
         Insp  s = new Insp();
            s.g= new Y();
              s.impls();  // line two
   }}
```

Choose only one answer:

a. Runtime Error
b. Compile Error at line two
c. Brr
d. Compile Error at line one

37)

```
class FireCracker {
    FireCracker f = new FireCracker();
    int x =39;
    int y =89;
   public static void main(String[] args) {
       FireCracker f;
       f= new FireCracker();        //line one
       System.out.print(f.x);       //line two
       System.out.print(f.y);       //line three

   }}
```

Choose only one answer:

a. 3989
b. Runtime Error at line one
c. Runtime Error at line two
d. Compile Error at line one
e. Compile Error at line three

38)

```
class Lio{}
class Cio extends Lio{
   void mr(){
      System.out.print("mar mar");
   }
}

class Bandlow {
   public static void main(String[] args) {
      Lio i = new Lio();
      Cio c = new Cio();
      Lio l = new Cio();
      i.mr();//line one
      c.mr();//line two
      l.mr();//line three

   }}
```

Choose two:

a. mar marmar marmar mar
b. Compile Error at line one
c. Compile Error at line two
d. Compile Error at line three
e. Runtime Error

39)

```
class A{
   int x =89;
 A m(A a){
    a.x=400;  // line one
    return a;//  line two}
public static void main(String[] args) {
    A a = new A();
    System.out.print(a.x);
      a.m(a);    //line three
    System.out.print(a.x);}}
```

Choose only one answer:

a. 89400
b. Compile Error at line one
c. Compile Error at line two
d. Compile Error at line three

40)

```
class A {
   int i = 10;
   public void printValue() {
      System.out.println("Value-A");
   };}
class B extends A {
   int i = 12;
   public void printValue() {
      System.out.print("Value-B");}}
class Test {
   public static void main(String argv[]) {
      A a = new B();
      a.printValue();
      System.out.println(a.i);}}
```

Choose only one answer:

a. Value-B 11
b. Value-B 10

c. Value-A 10
d. Value-A 11

41)

```
class Test {
   public static void main(String[] args) {
      String value = "abc";
      changeValue(value);
      System.out.println(value);
   }
   public static void changeValue(String a) {
      a = "xyz";
   }
}
```

Choose only one answer:

a. abc
b. xyz
c. Compilation fails
d. Compilation clean but no output

42)

```
class D {
   int i;
   int j;
   public D(int i, int j) {
      this.i = i;
      this.j = j;
   }
   public void printName() {
      System.out.println("Name-D");// line 01
   }
}
class Test {
   public static void main(String[] args) {
      D d = new D();  // Line 02
      d.printName();

   }
}
```

Choose only one answer:

a. Name-D
b. Compilation fails due to an error on lines 1
c. Compilation fails due to an error on lines 2
d. Compilation succeed but no output

43)

```
class Base {
        public final int getNext(int i) {
                return ++i;
}}
 class Derived extends Base {
        public int getNext(int i) {
                return i++;
        }
        public static void main(String[] args) {
                int result = new Derived().getNext(3);
                System.out.print(result);
                result = new Base().getNext(3);
                System.out.print(result);
}}
```

Choose only one answer:

a. 33
b. 34
c. 44
d. 43
e. a compilation error

44)

```
class Calculator {
   int num = 100;
   public void calc(int num) {
      this.num = num * 10;
   }
   public void printNum() {
      System.out.println(num);
   }
   public static void main(String[] args) {
      Calculator obj = new Calculator();
      obj.calc(2);
      obj.printNum();
   }
}
```

What is the result?

a. 20
b. 100
c. 1000
d. 2

45)

```
class Alpha {
   String getType() {
      return "alpha";
   }
}
class Beta extends Alpha {

   String getType() {
      return "beta";
   }
}
class Gamma extends Beta {
   String getType() {
      return "gamma";
   }
   public static void main(String[] args) {
      Gamma g1 = new Alpha();
      Gamma g2 = new Beta();
      System.out.println(g1.getType() + " "
            + g2.getType());
   }
}
```

What is the result?

a. alpha beta
b. beta beta
c. gamma gamma
d. Compilation fails.

46)

```
class Account{}
class Customer{Account ac;}
```

Which of the following statements are TRUE about the above code?

a. Choice 1 : Customer has a HAS-A relationship with Account.
b. Choice 2 : Customer has a IS-A relationship with Account.
c. Choice 3 : Account has a IS-A relationship with Customer.
d. Choice 4 : None

47)

```
class CleaningMachine{
  void clean(){
     System.out.println("fooooo");
  }
}
class VacuumCleaner extends CleaningMachine{
  void clean(){
    System.out.println("VacuumCleaner - Clean");
  }
}
class Cleaner {
   CleaningMachine c;

}
class CleaningMaster{
   public static void main(String[] args) {
         Cleaner c = new Cleaner();
               c.c = new VacuumCleaner();
               c.c.clean();
   }
}
```

What is the output: : VacuumCleaner - Clean

48)

```
public class Main{
     public int i;
     public static void main(String argv[]){
       Main sc = new Main();
       // Comment line
     }
   }
```

Which of the following statements are correct if they replace the comment line?

a. System.out.println(i);
b. System.out.println(sc.i);
c. System.out.println(Main.i);
d. System.out.println((new Main()).i);

49) What is the result of the following operation ?

```
System.out.println(4 % 3);
```

Answer : 1

50)

```
class Array {
   public static void main(String[] args) {
      int array[] = new int[]{13, 14, 15, 16, 17};
      for (int current_value = 0; current_value < array.length; current_value++) {
```

```
            System.out.print(array[current_value]);
        }
    }
}
```

a. 1314151616
b. 1314151515
c. Compile Error
d. 1314151617
e. RunTime Error

51)

```
class WhileLoop{
   public static void main(String []args){
        int x = 8;
        while (x > 8) {
            System.out.println("in the loop");
            x = 10;
         }
        System.out.println("past the loop");
   }
}
```

a. in the loop
b. in the loop
past the loop
c. past the loop
d. past the loop
in the loop
e. Compile Error

52)

```
class  ArrayExample{
public static void main(String []args){
    int load[][] ={{1,2},{3,4},{5,6}};
        for(int i=0;i< load.length;i++){
            for(int j=0;j<load[0].length;j++){
                System.out.println(load[i][j]);
                    }
                }
  }
}
```

a. 1
2
3
4
5
6

b. 1
2
3
4
5

c. 0
1

2
3
4

d. Compile Error
e. None Of the above

53)

```
class A {
   A() {
      System.out.println("A");
   }
   A(int x) {
      System.out.println("A int");
   }
}
class B extends A {
   B() {
      super(10);
   }
}
class Test {
  public static void main(String[] args) {
      B b = new B();
   }
}
```

a. A int , A
b. A , A int
c. Compile Error
d. A int

54) What is the output for the below code?

```
class A {
   int k;
   boolean istrue;
   static int p;
   public void printValue() {
      System.out.print(k);
      System.out.print(istrue);
      System.out.print(p);
   }
}
class Test {
   public static void main(String argv[]) {
      A a = new A();
      a.printValue();
   }
}
```

Choose only one answer:

a. 0 false 0
b. 0 true 0
c. 0 0 0
d. Compile error - static variable must be initialized before use.

55)

```
abstract class A {
   public abstract void m();  //Line 1
   public void n() {
      System.out.println("A n");
   }
}
abstract class B {
   public void m() { // Line 2
      System.out.println("B m");
   }
}
class Test {
   public static void main(String[] args) {
      B b = new B(); //Line 3
      b.m();
   }
}
```

a. B m
b. Compile Error at Line 1
c. Compile Error at Line 2
d. Compile Error at Line 3

56)

```
class A {
   A(){ //Line 1
      this();
   }
   A(int i){
      System.out.println("A int");
   }
}
class B{
   B(int x){
      super(); //Line 2
      System.out.println("B int");
   }
}
class Test {
```

```
    public static void main(String[] args) {
      A a = new A();
    }
}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. B int , A int
d. A int , B int

57)

```
abstract interface A{ //Line 1
  int x = 10; //Line 2
  static int y = 20;
  public void m();
}
class B implements A{
  public void m(){
    System.out.println("B m");
    System.out.println(A.y); //Line 3
  }
}
class Test{
  public static void main(String[] args) {
    B b = new B();
    b.m();
  }
}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. Compile Error at Line 3
d. B m , 20

58)

```
class A {
  A() {
    System.out.println("A");
  }
  A(String a) {
    System.out.println(a);
  }
}
class B extends A {
  B() {
   super("B"); //Line 1
  }
```

```
    A a = new A(); //Line 2
}
class Test{
    public static void main(String[] args) {
        B b = new B();
    }
}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. B , A
d. A , B

59)

```
interface A {
    public abstract void m();
}
interface B {
    public abstract void n();
}
class C implements A, B { //Line 1
    public void m() {
        System.out.println("m");
    }
    public void n() {
        System.out.println("n");
    }
}
class Test {
    public static void main(String[] args) {
        A b = new C(); //Line 2
        b.m(); //Line 3
    }
}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. Compile Error at Line 3
d. m

60)

```
abstract class A {
    A() { //Line 1
        System.out.println("A");
    }
}
```

```
class B extends A {
   B() {
      super(); //Line 2
      System.out.println("B");
   }
   B(int x) {
      System.out.println(x);
   }
}
class C extends B {
   public static void main(String[] args) {
      C c = new C();
      B b = new B(10);
   }
}
```

a. Compile Error at Line 1
b. Compile Error at Line 2
c. A , B , A , 10
d. A , B , 10

Section B

1. Write a program
   1.1. Write a program to create a class named **shape.**
   1.2. In this class we have three sub classes **circle, triangle** and **square**
   1.3. Each class has two member functions named **draw ()** and **erase ()**. Create these using polymorphism concepts.
   1.4. And write a **test class** to Demonstrate the function of Triangle class.

2. Write a program to give the example for method overriding concepts.
3. Write a program to demonstrate 1D ,2D and 3D Arrays .
4. what are the differences between overriding and Overloading.
5. What are the garbage Collection Types. Describe with Code Examples.

Section B Answers :

Q 01 :

```
class Shape {
   void draw() {System.out.println("Draw");
   }
   void erase() {System.out.println("Erase");
   }
}
class Circle extends Shape {
}
```

```
class Triangle extends Shape {
}
class Square extends Shape {
}
class Test{public static void main(String[] args) {
        Triangle t=new Triangle();
        t.draw();
        t.erase();
   }
}
```

Q2 :

•Argument list must exactly match .

```
class A {
void m(int A){
            }
}
class B extends A{
void m(int B){
                        }
}
```

•Same or wider access level

```
class A {

    public void m(){

      }}

class B extends A{

    //  void m(){} == Compile Error

     public void m(){} // == Compile because Access level public is same

}
```

•You cannot override a method marked final.

```
class A {
            final void m(int A) {
            }
}
class B extends A {
            void m(int A) {
            }
            public static void main(String[] args) {
            }
}
```

•You cannot override a method marked static.

```
class A {
```

```
        static void m(int i) {
        System.out.println("A");
        }
}
class B extends A {
        static void m(int i) {
        System.out.println("B");
        }
        public static void main(String[] args) {
        B b = new B();
        }
}
```

Q3 .

```
class A {

   public static void main(String[] args) {
      int i1[] = new int[3];  // == 1D Array
      i1[0] = 10;
      i1[1] = 20;
      i1[2] = 30;
      int i2[][] = new int[2][2];    // == 2D Array
      i2[0][1] = 40;
      int i3[][][] = new int[2][2][2];    // == 3D Array
      i3[0][0][1] = 50;
      System.out.println(i1[0]);
      System.out.println(i1[1]);
      System.out.println(i2[0][1]);
      System.out.println(i3[0][0][1]);
   }
}
```

Q4.

Overriding	Overloading
Method overloading is used *to increase the readability* of the program.	Method overriding is used *to provide the specific implementation* of the method that is already provided by its super class.
Method overloading is performed *within class*.	Method overriding occurs *in two classes* that have IS-A (inheritance) relationship.
In case of method overloading, *parameter must be different*.	In case of method overriding, *parameter must be same*.
Method overloading is the example of *compile time polymorphism*.	Method overriding is the example of *run time polymorphism*.

Return type can be same or different in method overloading. But you must have to change the parameter.	*Return type must be same or covariant* in method overriding.

Q5 .

- Nulling Reference
- Reassigning
- local
- Isolated Island

- **Nulling Reference :**

```
class Garbage {
   public static void main(String[] args) {
      Garbage b1 = new Garbage();
      System.out.println(b1);
      b1 = null;
      System.out.println(b1);
   }
}
```

- **Reassigning :**

```
class Garbage {

   public static void main(String[] args) {

      Garbage b1 = new Garbage();

      System.out.println(b1);

      b1 = new Garbage();

      System.out.println(b1);

   }

}
```

- **Local**

```
class Garbage {
   static void m() {
      Garbage b1 = new Garbage();
      System.out.println(b1);
   }
   public static void main(String[] args) {
      m();
   }
}
```

- **Isolated Island**

```
 class A {
A a;
            public static void main(String[] args) {
            A a1=new A();
            A a2=new A();
            A a3=new A();
         a1.a=a2;
         a2.a=a3;
         a3.a=a1;
         a1=null;
         a2=null;
         a3=null;
            }
}
```