1)

// Main.java

public class Main

{

    public static void gfg(String s)

    {

        System.out.println("String");

    }

    public static void gfg(Object o)

    {

        System.out.println("Object");

    }

    public static void main(String args[])

    {

        gfg(null);

    }

} //end class

2)

// Main.java

public class Main

{

    public static void gfg(String s)

    {

        System.out.println("String");

    }

    public static void gfg(Object o)

    {

        System.out.println("Object");

    }

    public static void gfg(Integer i)

    {

        System.out.println("Integer");

    }

    public static void main(String args[])

    {

        gfg(null);

    }

} //end class

|  |
| --- |
| **3)Important**  // Main.java  public class Main  {      public static void main(String args[])      {          String s1 = "abc";          String s2 = s1;          s1 += "d";          System.out.println(s1 + " " + s2 + " " + (s1 == s2));            StringBuffer sb1 = new StringBuffer("abc");          StringBuffer sb2 = sb1;          sb1.append("d");          System.out.println(sb1 + " " + sb2 + " " + (sb1 == sb2));      }  } //end class |

|  |  |
| --- | --- |
| 4)class First  {      public First() {  System.out.println("a"); }  }    class Second extends First  {      public Second()  {  System.out.println("b"); }  }    class Third extends Second  {      public Third()   {  System.out.println("c"); }  }    public class MainClass  {      public static void main(String[] args)      {          Third c = new Third();      }  } | |
| 5) class First  {      int i = 10;        public First(int j)      {          System.out.println(i);          this.i = j \* 10;      }  }    class Second extends First  {      public Second(int j)      {          super(j);          System.out.println(i);          this.i = j \* 20;      }  }    public class MainClass  {      public static void main(String[] args)      {          Second n = new Second(20);          System.out.println(n.i);      }  } |
|  |
| 6)  import java.util.\*;  class I  {      public static void main (String[] args)      {          Object i = new ArrayList().iterator();          System.out.print((i instanceof List) + ", ");          System.out.print((i instanceof Iterator) + ", ");          System.out.print(i instanceof ListIterator);      }  } |

7)

|  |
| --- |
| class ThreadEx extends Thread  {      public void run()      {          System.out.print("Hello...");      }      public static void main(String args[])      {          ThreadEx T1 = new ThreadEx();          T1.start();          T1.stop();          T1.start();      }  } |
| 8)  public 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();      }  } | |

9)

|  |
| --- |
| public class MyStuff  {      String name;        MyStuff(String n) {  name = n;  }        public static void main(String[] args)      {          MyStuff m1 = new MyStuff("guitar");          MyStuff m2 = new MyStuff("tv");          System.out.println(m2.equals(m1));      }        @Override      public boolean equals(Object obj)      {          MyStuff m = (MyStuff) obj;          if (m.name != null)  { return true;  }          return false;      }  } |

**Options :**  
A) The output is true and MyStuff fulfills the Object.equals() contract.  
B) The output is false and MyStuff fulfills the Object.equals() contract.  
C) The output is true and MyStuff does NOT fulfill the Object.equals() contract.  
D) The output is false and MyStuff does NOT fulfill the Object.equals() contract.

10)

|  |
| --- |
| class Alpha  {      public String type = "a ";      public Alpha() {  System.out.print("alpha "); }  }    public class Beta extends Alpha  {      public Beta()  {  System.out.print("beta ");  }        void go()      {          type = "b ";          System.out.print(this.type + super.type);      }        public static void main(String[] args)      {          new Beta().go();      }  } |

**Options :**  
A) alpha beta b b  
B) alpha beta a b  
C) beta alpha b b  
D) beta alpha a b

|  |
| --- |
| 11)public class Test  {      public static void main(String[] args)      {          StringBuilder s1 = new StringBuilder("Java");          String s2 = "Love";          s1.append(s2);          s1.substring(4);          int foundAt = s1.indexOf(s2);          System.out.println(foundAt);      }  } |

**Options :**  
A) -1  
B) 3  
C) 4  
D) A **StringIndexOutOfBoundsException** is thrown at runtime.

|  |
| --- |
| 12)class Writer  {      public  static void write()      {          System.out.println("Writing...");      }  }  class Author extends Writer  {      public  static void write()      {          System.out.println("Writing book");      }  }    public class Programmer extends Author  {      public  static void write()      {          System.out.println("Writing code");      }        public static void main(String[] args)      {          Author a = new Programmer();          a.write();      }  } |

**Options :**  
A) Writing…  
B) Writing book  
C) Writing code  
D) Compilation fails

|  |
| --- |
| 13)class GfG  {      public static void main(String args[])      {          String s1 = new String("geeksforgeeks");          String s2 = new String("geeksforgeeks");          if (s1 == s2)              System.out.println("Equal");          else              System.out.println("Not equal");      }  } |

14)

|  |
| --- |
| class Person  {      private void who()      {          System.out.println("Inside private method Person(who)");      }        public static void whoAmI()      {          System.out.println("Inside static method, Person(whoAmI)");      }        public void whoAreYou()      {          who();          System.out.println("Inside virtual method, Person(whoAreYou)");      }  }    class Kid extends Person  {      private void who()      {          System.out.println("Kid(who)");      }        public static void whoAmI()      {          System.out.println("Kid(whoAmI)");      }        public void whoAreYou()      {          who();          System.out.println("Kid(whoAreYou)");      }  }  public class Gfg  {      public static void main(String args[])      {          Person p = new Kid();          p.whoAmI();          p.whoAreYou();      }  } |

15)

|  |
| --- |
| class One implements Runnable  {      public void run()      {          System.out.print(Thread.currentThread().getName());      }  }  class Two implements Runnable  {      public void run()      {          new One().run();          new Thread(new One(),"gfg2").run();          new Thread(new One(),"gfg3").start();      }  }  class Three  {      public static void main (String[] args)      {          new Thread(new Two(),"gfg1").start();      }  } |

17)class Gfg

{

// constructor

Gfg(int a)

{

System.out.println("Geeksforgeeks"+a);

}

static Gfg a = new Gfg(1); //line 8

public static void main(String args[])

{

Gfg b; //line 12

b = new Gfg(2);

}

}

18)

|  |
| --- |
| class Gfg  {      static int num;      static String mystr;        // constructor      Gfg()      {          num = 100;          mystr = "Constructor";      }        // First Static block      static      {          System.out.println("Static Block 1");          num = 68;          mystr = "Block1";      }        // Second static block      static      {          System.out.println("Static Block 2");          num = 98;          mystr = "Block2";      }        public static void main(String args[])      {          Gfg a = new Gfg();          System.out.println("Value of num = " + a.num);          System.out.println("Value of mystr = " + a.mystr);      }  } |

19)

|  |
| --- |
| class superClass  {      final public int calc(int a, int b)      {          return 0;      }  }  class subClass extends superClass  {      public int calc(int a, int b)      {          return 1;      }  }  public class Gfg  {      public static void main(String args[])      {          subClass get = new subClass();          System.out.println("x = " + get.calc(0, 1));      }  } |
| 20)public class Gfg  {      public static void main(String[] args)      {    Integer a = 128, b = 128;          System.out.println(a == b);            Integer c = 100, d = 100;          System.out.println(c == d);      }  } |

21)

|  |
| --- |
| public class Test  {      public static void main(String[] args) throws InterruptedException      {          String str = new String("GeeksForGeeks");            // making str eligible for gc          str = null;            // calling garbage collector          System.gc();            // waiting for gc to complete          Thread.sleep(1000);            System.out.println("end of main");      }        @Override      protected void finalize()      {          System.out.println("finalize method called");      }  } |

**22)**

|  |
| --- |
| public class Test  {      public static void main(String[] args) throws InterruptedException      {          Test t = new Test();            // making t eligible for garbage collection          t = null;            // calling garbage collector          System.gc();            // waiting for gc to complete          Thread.sleep(1000);            System.out.println("end main");      }        @Override      protected void finalize()      {          System.out.println("finalize method called");          System.out.println(10/0);      }    } |

23)

|  |
| --- |
| public class Test  {      public static void main(String[] args)      {          // How many objects are eligible for          // garbage collection after this line?          m1();  // Line 5      }        static void m1()      {          Test t1 = new Test();          Test t2 = new Test();      }  } |

* **Question :**  
  How many objects are eligible for garbage collection after execution of line 5 ?

**24)**

|  |
| --- |
| public class Base  {      private int data;        public Base()      {          data = 5;      }        public int getData()      {          return this.data;      }  }    class Derived extends Base  {      private int data;      public Derived()      {          data = 6;      }      private int getData()      {          return data;      }        public static void main(String[] args)      {          Derived myData = new Derived();          System.out.println(myData.getData());      }  } |

a) 6  
b) 5  
c) Compile time error  
d) Run time error

25)

|  |
| --- |
| public class Test  {      private int data = 5;        public int getData()      {          return this.data;      }      public int getData(int value)      {          return (data+1);      }      public int getData(int... value)      {          return  (data+2);      }        public static void main(String[] args)      {          Test temp = new Test();          System.out.println(temp.getData(7, 8, 12));      }  } |

a) Either Compile time or Runtime error  
b) 8  
c) 10  
d) 7

26)

|  |
| --- |
| public class Base  {      private int multiplier(int data)      {          return data\*5;      }  }    class Derived extends Base  {      private static int data;      public Derived()      {          data = 25;      }      public static void main(String[] args)      {          Base temp = new Derived();          System.out.println(temp.multiplier(data));      }  } |

a) 125  
b) 25  
c) Runtime error  
d) Compile time error

|  |
| --- |
| 27)  import java.io.IOException;  import java.util.EmptyStackException;    public class newclass  {      public static void main(String[] args)      {          try          {              System.out.printf("%d", 1);              throw(new Exception());          }          catch(IOException e)          {              System.out.printf("%d", 2);          }          catch(EmptyStackException e)          {              System.out.printf("%d", 3);          }          catch(Exception e)          {              System.out.printf("%d", 4);          }          finally          {              System.out.printf("%d", 5);          }      }  } |

a) 12345  
b) 15  
c) 135  
d) 145

28)

|  |
| --- |
| public class javaclass  {      static      {          System.out.printf("%d", 1);      }      static      {          System.out.printf("%d", 2);      }      static      {          System.out.printf("%d", 3);      }      private static int myMethod()      {          return 4;      }      private int function()      {          return 5;      }        public static void main(String[] args)      {          System.out.printf("%d", (new javaclass()).function() + myMethod());      }  } |

a) 123  
b) 45  
c) 12345  
d) 1239

|  |
| --- |
| 29)public class Test  {      private static int value = 20;      public int s = 15;      public static int temp = 10;      public static class Nested      {        private void display()        {            System.out.println(temp + s + value);        }      }        public static void main(String args[])      {        Test.Nested inner = new Test.Nested();        inner.display();      }  } |

a) Compilation error  
b) 1020  
c) 101520  
d) None of the above

30)

|  |
| --- |
| import java.io.\*;  public class Test  {      public void display() throws IOException      {          System.out.println("Test");      }  }  class Derived extends Test  {      public void display() throws IOException      {          System.out.println("Derived");      }      public static void main(String[] args) throws IOException      {          Derived object = new Derived();          object.display();      }  } |

a) Test  
b) Derived  
c) Compilation error  
d) Runtime error

31)

|  |
| --- |
| public class Test extends Thread  {      public void run()      {          System.out.printf("Test ");      }      public static void main(String[] args)      {          Test test = new Test();          test.run();          test.start();      }  } |

a) Compilation error  
b) Runtime error  
c) Test  
d) Test Test

32)

|  |
| --- |
| public class Test extends Thread  {      public static void main(String[] args)      {          String a = "GeeksforGeeks";          String b = new String(a);          int value = 0;          value = (a==b) ? 1:2;          if(value == 1)          {              System.out.println("GeeksforGeeks");          }          else if(value == 2)          {              System.out.println("Geeks for Geeks");          }          else          {              System.out.println("GFG");          }        }  } |

a) GeeksforGeeks  
b) Geeks for Geeks  
c) GFG  
d) None of the above

|  |
| --- |
| 33)public class Test  {      try      {          public Test()          {              System.out.println("GeeksforGeeks");              throw new Exception();          }      }      catch(Exception e)      {          System.out.println("GFG");      }      public static void main(String[] args)      {          Test test = new Test();      }  } |

a) GeeksforGeeks  
b) GFG  
c) Compilation error  
d) None of the above

34)

|  |
| --- |
| import java.util.\*;    public class priorityQueue  {      public static void main(String[] args)      {          PriorityQueue<Integer> queue =                              new PriorityQueue<>();          queue.add(11);          queue.add(10);          queue.add(22);          queue.add(5);          queue.add(12);          queue.add(2);            while (queue.isEmpty() == false)              System.out.printf("%d ", queue.remove());            System.out.println("\n");      }  } |

a) 11 10 22 5 12 2  
b) 2 12 5 22 10 11  
c) 2 5 10 11 12 22  
d) 22 12 11 10 5 2

**35)**

|  |
| --- |
| import java.util.\*;    public class Treeset  {      public static void main(String[] args)      {          TreeSet<String> treeSet = new TreeSet<>();            treeSet.add("Geeks");          treeSet.add("for");          treeSet.add("Geeks");          treeSet.add("GeeksforGeeks");            for (String temp : treeSet)              System.out.printf(temp + " ");            System.out.println("\n");      }  } |

a) Geeks for Geeks GeeksforGeeks  
b) Geeks for GeeksforGeeks  
c) Geeks GeeksforGeeks for  
d) for GeeksforGeeks Geeks

**36)**

|  |
| --- |
| import java.util.\*;    public class linkedList  {      public static void main(String[] args)      {          List<String> list1 = new LinkedList<>();          list1.add("Geeks");          list1.add("for");          list1.add("Geeks");          list1.add("GFG");          list1.add("GeeksforGeeks");            List<String> list2 = new LinkedList<>();          list2.add("Geeks");            list1.removeAll(list2);            for (String temp : list1)              System.out.printf(temp + " ");            System.out.println();      }  } |

a) for Geeks GFG GeeksforGeeks  
b) for GeeksforGeeks GFG  
c) for GFG for  
d) for GFG GeeksforGeeks