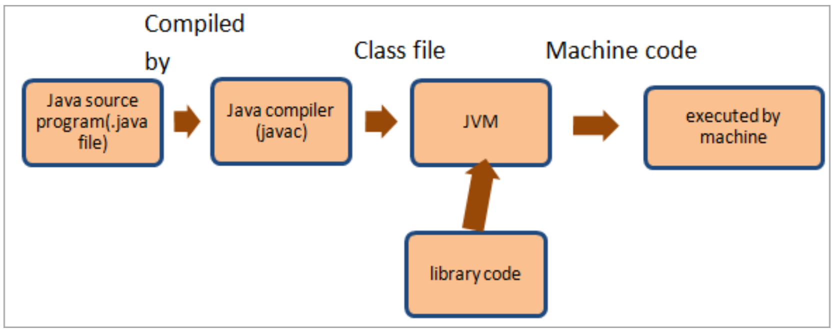
Statements written with red are the answers of the questions and statements written in orange are explanations.

Question a- **Features/Characteristics of Java**

1- Which of the followings is not a feature of Java Language?
a- Architecture Neutral
b- Portable
c- Robust
d- Multithreaded
e- Use of pointers

Question b- **JVM, JRE, JDK, Common errors**

1- Draw the diagram of the "flow of a Java program".



2- Which of the codes will cause a logic error?

```
a-  public void getName()
 {  return this.name;
 } //compile time error
b- String[] arr = {"Elif","Ebru","Kevser","Bilgesu"};
      for(int i=0;i<=arr.length;i++) {
          System.out.println(arr[i]);
      } //It will throw ArrayIndexOutOfBoundException at iteration 4 →   (runtime)

c-   public static void main(String[] args)
  {
     int c;
     for (c=1; c<=10; c++) ;
     {
       System.out.println("Count is " +c);
     }
  }
 //logic error → due to misplaced semicolon


d-public static void main(String args[])
   {
      int a = 4, b = 6;
```

```
        int Sum = a + b;
        System.out.println("Sum of variables is " + sum);
    } //Sum vs sum → compile time error

e- public static void main(String args[])
    {
        int a = 7, answer;
        int i;
        for (i = 1, i <= 10; i++) {
            answer = a * i;
            System.out.println(answer);
        }
    }      // Should have been for(i=1; i<=10; i++) → compile time
```

Question c- Casting

Which one of the following codes work?

```
a-   int   it = 120;
     float ft;
     ft = it;
```

```
b-  int   it;

    float ft = 3.1444f;

    it = ft; // "Explicit cast needed to convert float to int."
```

```
c-  double   db = 3.8644951;

    float    ft;

    ft = db;
```

```
d-   float ft = 3.8444f;

     float sum;

     sum = 2.0 + ft;

      sum = (float) (2.0 + ft);
```

//This will not work because 2.0 on the right-hand-side of the last expression is a double-precision number, so "ft" is first converted to a double-precision number before it is added to "2.0", then right-hand-side is of type "double" while the left- hand-side is of type "float"

Question d- Constructors

True – False

- We can define a constructor in Interface -F
- There are three kinds of constructors in Java -F (2: Default and Parameterized)
- Constructors cannot be static -T
- Constructors can be declared as protected -T

Question e- Inheritance and polymorphism

1- Choose the true statements

a- A class can extend itself

b- The subclass inherits fields and methods from the superclass without any of them having to be rewritten.

c- For the subclass to inherit fields and methods from the superclass, they need to be overridden.

d- Superclass constructors are not inherited (Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass)

e- Method Overloading is runtime polymorphism (method overriding is runtime polymorphism)

Question f- Loops, break, continue, if-else

1- Will this code compile, if your answer is no, explain the reason.

```
public class Example {
  public static void main(String args[]){
     for (int i=0; i<=6; i++)
     {
        if (i==4)
        {
          continue;
          System.out.print("Inside of the if statement ");
        }
        System.out.print(i+" ");
     }
  }
}
```

No. It gives an unreachable statement error. System.out.print(" Inside the if statement "); cannot be reached.

Question g- Abstract class and interfaces

True- False

- Abstract class must have only abstract methods. -F
- Interface can have only abstract, default and static methods. -T (Since Java 8)
- Interface can provide the implementation of abstract class. -F
- Abstract class can have final, non-final, static and non-static variables. -T

Question h- Exception handling and final keyword

Can the following codes be compiled? Why? :

a- This code will compile:

```
try {

}
catch (Exception e) {

}
catch (ArithmeticException a) {
}
```

It cannot. This first handler catches exceptions of type Exception so it catches any exception, including ArithmeticException. The second handler could never be reached.

b-

```
try {

 throw new Exception();

}

System.out.println("Between try and catch blocks");

catch(Exception e) { }
```

It cannot. We can not put any statements between try-catch blocks.

c-

```
try{
        int num=1/0;
        System.out.println(num);
    }
    finally{
        System.out.println("This is final block");
    }
```

It can. A try statement does not have to have a catch block if it has a finally block.

Question i- Threads

Give three differences between Runnable and Callable interfaces.

Runnable:

1- It cannot be passed to invokeAll method.
2- Cannot throw a checked exception.
3- Uses the run() method to define a task

Callable:

1- It can be passed to invokeAll method.
2- It can throw an exception.
3- It uses the call() method to define a task.

Question j- Synchronization

Why must wait() method be called from the synchronized block?

java.lang.IllegalMonitorStateException exception will be thrown if wait method is not called.