```
public class Wall {
    public static void main(String args[]) {
        double area = 10.98;
        String color;
        if (area < 5)
            color = "red";
        else
            color = "blue";
        System.out.println(color);
    }
}
```

- Was ist die Ausgabe? (1)

A. red

B. blue

C. No output

D. Compilation error

```
interface Jumpable {}
class Animal {}
class Lion extends Animal implements Jumpable {}
```

- Welche Anweisungen sind verursachen keine Fehler? (2)

A. `Jumpable var1 = new Jumpable();`

B. `Animal var2 = new Animal();`

C. `Lion var3 = new Animal();`

D. `Jumpable var4 = new Animal();`

E. `Jumpable var5 = new Lion();`

F. `Jumpable var6 = (Jumpable)(new Animal());`

```
try {
    String[][] names = {{"Andre", "Mike"}, null,
{"Pedro"}};
    System.out.println (names[2][1].substring(0, 2));
} catch (/* CODE */) {
    System.out.println(1);
}
```

- Was kann eingesetzt werden, damit in der Ausgabe 1 erfolgt? (1)

A. `IndexPositionException e`

B. `NullPointerException e`

C. `ArrayIndexOutOfBoundsException e`

D. `ArrayOutOfBoundsException e`

```
import java.time.*;
class Camera {
    public static void main(String args[]) {
        int hours;
        LocalDateTime now = LocalDateTime.of(2020, 10, 01, 0 , 0);
        LocalDate before = now.toLocalDate().minusDays(1);
        LocalTime after = now.toLocalTime().plusHours(1);
        while (before.isBefore(after) && hours < 4) {
            ++hours;
        }
        System.out.println("Hours:" + hours);
    }
}
```

- Was ist die Ausgabe? (1)

A. The code prints Camera:null.

B. The code prints Camera:Adjust settings manually.

C. The code prints Camera:.

D. The code will fail to compile.

```
interface Runner {
  public void run();
}
```

- Which of the following is/are valid lambda expression(s) that capture(s) the above interface? (2)

```
A.  -> System.out.println("running...");

B. void -> System.out.println("running...")

C. () -> System.out.println("running...")

D. () -> { System.out.println("running..."); return; }

E. (void) -> System.out.println("running...")

F.  -> System.out.println("running...")
```

```
class Student { int marks = 10; }

class Result {
    public static void main(String... args) {
        Student s = new Student();
        switch (s.marks) {
            default: System.out.println("100");
            case 10: System.out.println("10");
            case 98: System.out.println("98");
        }
    }
}
```

- Was ist die Ausgabe? (1)

A.
```
100
10
98
```

B.
```
10
98
```

C.
```
100
```

D.
```
10
```

```
class Pencil {}
class ColorPencil extends Pencil {
    String color;
    ColorPencil(String color) {this.color = color;}
}
```

- Wie kann man ColorPencil initialisieren? (2)

A. `ColorPencil var1 = new ColorPencil();`

B. `ColorPencil var2 = new ColorPencil(RED);`

C. `ColorPencil var3 = new ColorPencil("RED");`

D. `Pencil var4 = new ColorPencil("BLUE");`

```
class Doctor {
    protected int age;
    protected void setAge(int val) { age = val; }
    protected int getAge() { return age; }
}

class Surgeon extends Doctor {
    Surgeon(String val) {
        specialization = val;
    }
    String specialization;
    String getSpecialization() { return specialization; }
}

class Hospital {
    public static void main(String args[]) {
        Surgeon s1 = new Surgeon("Liver");
        Surgeon s2 = new Surgeon("Heart");
        s1.age = 45;
        System.out.println(s1.age + s2.getSpecialization());
        System.out.println(s2.age + s1.getSpecialization());
    }
}
```

- Wie ist die Ausgabe? (1)

A.
```
45Heart
0Liver
```

B.
```
45Liver
0Heart
```

C.
```
45Liver
45Heart
```

D.
```
45Heart
45Heart
```

E. Class fails to compile.

```
import java.util.*;
class EJGArrayL {
    public static void main(String args[]) {
        ArrayList<String> seasons = new ArrayList<>();
        seasons.add(1, "Spring"); seasons.add(2, "Summer");
        seasons.add(3, "Autumn"); seasons.add(4, "Winter");
        seasons.remove(2);

        for (String s : seasons)
            System.out.print(s + ", ");
    }
}
```

- Was ist die Ausgabe? (1)

A. Spring, Summer, Winter,

B. Spring, Autumn, Winter,

C. Autumn, Winter,

D. Compilation error

E. Runtime exception

```
class RocketScience {
    public static void main(String args[]) {
        int a = 0;
        while (a == a++) {
            a++;
            System.out.println(a);
        }
    }
}
```

- Wie ist die Ausgabe? (1)

A. The while loop won't execute; nothing will be printed.

B. loop execute indefinitely, printing all numbers, starting from 1.

C. loop execute indefinitely, printing all even numbers, starting from 0.

D. loop execute indefinitely, printing all even numbers, starting from 2.

E. loop execute indefinitely, printing all odd numbers, starting from 1.

F. loop execute indefinitely, printing all odd numbers, starting from 3.

```
int a = 10;
for (; a <= 20; ++a) {
    if (a%3 == 0) a++; else if (a%2 == 0) a=a*2;
    System.out.println(a);
}
```

- Was ist die Ausgabe? (1)

A.
```
11
13
15
17
19
```

B.
```
20
```

C.
```
11
14
17
20
40
```

D. Compilation error

```
class Phone {
    void call() {
        System.out.println(„Call-Phone");
    }
}

class SmartPhone extends Phone{
    void call() {
        System.out.println("Call-SmartPhone");
    }
}
class TestPhones {
    public static void main(String[] args) {
        Phone phone = new Phone();
        Phone smartPhone = new SmartPhone();
        phone.call();
        smartPhone.call();
    }
}
```

- Was ist die Ausgabe? (1)

A.
```
Call-Phone
Call-Phone
```

B.
```
Call-Phone
Call-SmartPhone
```

C.
```
Call-Phone
null
```

D.
```
null
Call-SmartPhone
```

```
1> class StringBuilders {
2>     public static void main(String... args) {
3>         StringBuilder sb1 = new StringBuilder("eLion");
4>         String ejg = null;
5>         ejg = sb1.append("X").substring(sb1.indexOf("L"),
sb1.indexOf("X"));
6>         System.out.println(ejg);
7>     }
8> }
```

- Wähle die korrekte Aussage aus (1)

A. The code will print LionX.

B. The code will print Lion.

C. The code will print Lion if line 5 is changed to the following:
```
ejg = sb1.append("X").substring(sb1.indexOf('L'),
sb1.indexOf('X'));
```

D. The code will compile only when line 4 is changed to the following:
```
StringBuilder ejg = null;
```

```
com.ejava is a package
class Person is defined in package com.ejava
class Course is defined in package com.ejava
```

- Richtiger import der Klassen Person und Course in die Klasse MyEJava? (3)

A.
```
import com.ejava.*;
class MyEJava {}
```

B.
```
import com.ejava;
class MyEJava {}
```

C.
```
import com.ejava.Person;
import com.ejava.Course;
class MyEJava {}
```

D.
```
import com.ejava.Person;
import com.ejava.*;
class MyEJava {}
```

```
1 class Animal {
2        public void printKing() {
3            System.out.println("Lion");
4        }
5    }
6    class Forest {
7        public static void main(String... args) {
8            Animal anAnimal = new Animal();
9            anAnimal.printKing();
10        }
11    }
```

- Wähle die richtigen Aussagen aus (2)

A. The class Forest prints Lion.

B. If the code on line 2 is changed as follows, the class Forest will print Lion:
   private void printKing() {

C. If the code on line 2 is changed as follows, the class Forest will print Lion:
   void printKing() {

D. If the code on line 2 is changed as follows, the class Forest will print Lion:
   default void printKing() {

```
class Phone {
    String keyboard = "in-built";
}
class Tablet extends Phone {
    boolean playMovie = false;
}
class College2 {
    public static void main(String args[]) {
        Phone phone = new Tablet();
        System.out.println(phone.keyboard + ":" + phone.playMovie);
    }
}
```

- Was ist die Ausgabe? (1)

A. `in-built:false`

B. `in-built:true`

C. `null:false`

D. `null:true`

E. `Compilation error`

```
class MainMethod {
    public static void main(String... args) {
         System.out.println(args[0]+":"+ args[2]);
    }
}
```

- Was ist die Ausgabe wenn mit folgender Zeile ausgefüht? (1)
  java MainMethod 1+2 2*3 4-3 5+1

A. java:1+2

B. java:3

C. MainMethod:2*3

D. MainMethod:6

E. 1+2:2*3

F. 3:3

G. 6

H. 1+2:4-3

I. 31

J. 4

```
String ejgStr[] = new String[][]{{null},new String[]
{"a","b","c"},{new String()}}[0] ;
String ejgStr1[] = null;
String ejgStr2[] = {null};
System.out.println(ejgStr[0]);
System.out.println(ejgStr2[0]);
System.out.println(ejgStr1[0]);
```

- Was ist die Ausgabe? (1)

A.
```
null
NullPointerException
```

B.
```
null
null
NullPointerException
```

C.
```
NullPointerException
```

D.
```
null
null
null
```

```
interface Moveable {
    int move(int distance);
}

class Person {
    static int MIN_DISTANCE = 5;
    int age;
    float height;
    boolean result;
    String name;
}

public class EJava {
    public static void main(String arguments[]) {
        Person person = new Person();
        Moveable moveable = (x) -> Person.MIN_DISTANCE + x;
        System.out.println(person.name + person.height + person.result
                                       + person.age + moveable.move(20));
    }
}
```

- Was ist die Ausgabe? (1)

  A. null0.0false025

  B. null0false025

  C. null0.0ffalse025

  D. 0.0false025

  E. 0false025

  F. 0.0ffalse025

  G. null0.0true025

  H. 0true025

  I. 0.0ftrue025

  J. Compilation error

  K. Runtime exception

```
class Printer {
    int inkLevel;
}

class LaserPrinter extends Printer {
    int pagesPerMin;
    public static void main(String args[]) {
        Printer myPrinter = new LaserPrinter();
        System.out.println(/* CODE */);
    }
}
```

- Was muss eingesetzt werden, damit der Wert von pagesPerMin ausgegeben wird (1)

A. `(LaserPrinter)myPrinter.pagesPerMin`

B. `myPrinter.pagesPerMin`

C. `LaserPrinter.myPrinter.pagesPerMin`

D. `((LaserPrinter)myPrinter).pagesPerMin`

```
class Bottle {
    void Bottle() {}
    void Bottle(WaterBottle w) {}
}
class WaterBottle extends Bottle {}
```

- Wähle die korrekten Aussagen (2)

A. A base class can't pass reference variables of its defined class as method parameters in constructors.

B. The class compiles successfully—a base class can use reference variables of its derived class as method parameters.

C. The class Bottle defines two overloaded constructors.

D. The class Bottle can access only one constructor.

```
interface Keys {
    String keypad(String region, int keys);
}

public class Handset {
    public static void main(String... args) {
        double price;
        String model;
        Keys varKeys = (region, keys) -> {
              if (keys >= 32)
                  return region;
              else
                  return „default";
        };
        System.out.println(model + price + varKeys.keypad("AB", 32));
     }
}
```

- Was ist die Ausgabe? (1)

    A. „null0AB

    B. null0.0AB

    C. null0default

    D. null0.0default

    E. 0

    F. 0.0

    G. Compilation error

```
public class Sales {
    public static void main(String args[]) {
        int salesPhone = 1;
        System.out.println(salesPhone++ + ++salesPhone +
++salesPhone);
    }
}
```

- Was ist die Ausgabe? (1)

A. 5

B. 6

C. 8

D. 9

- Welche Klasse kompiliert ? (1)

A.
```
package com.ejava.guru;
package com.ejava.oracle;
class MyClass {
    int age = /* 25 */ 74;
}
```

B.
```
import com.ejava.guru.*;
import com.ejava.oracle.*;
package com.ejava;
class MyClass {
    String name = "e" + "Ja /*va*/ v";
}
```

C.
```
class MyClass {
    import com.ejava.guru.*;
}
```

D.
```
class MyClass {
    int abc;
    String course = //this is a comment
    "eJava";
}
```

E. None of the above

```
public class TestClass{
   public static boolean checkList(List list, Predicate<List> p)
{
      return p.test(list);
   }
   public static void main(String[] args) {
      boolean b = //CODE
      System.out.println(b);
   }
}
```

- What can be inserted in the code below so that it will print true when run? (2)

A. `checkList(new ArrayList(), al -> al.isEmpty());`

B. `checkList(new ArrayList(), ArrayList al -> al.isEmpty());`

C. `checkList(new ArrayList(), al -> return al.size() == 0);`

D. `checkList(new ArrayList(), al -> al.add("hello"));`

E. `checkList(new ArrayList(), (ArrayList al) -> al.isEmpty());`

```
class EJava {
    // LINE 1
    public EJava() {
        System.out.println(name);
    }
    void calc() {
        // LINE 2
        if (8 > 2) {
            System.out.println(name);
        }
    }
    public static void main(String... args) {
        // LINE 3
        System.out.println(name);
    }
}
```

- Wähle eine angemessene Definition von name und die Zeile, so dass die Code kompiliert ? (1)

A. `Define static String name; on line 1.`

B. `Define String name; on line 1.`

C. `Define String name; on line 2.`

D. `Define String name; on line 3.`

```
package x;
public class TestClass {
    ArrayList<String> al;
    public void init(){
        al = new ArrayList<>();
        al.add("Name 1");
        al.add("Name 2");
    }
    public static void main(String[] args) throws Exception {
        TestClass tc = new TestClass();
        tc.init();
        System.out.println("Size = "+tc.al.size());
    }
}
```

- Which import statement should be added to make it compile? (1)

A. `import java.lang.*;`

B. `import java.lang.ArrayList;`

C. `import java.util.ArrayList;`

D. `import java.collections.ArrayList;`

E. `No import is necessary.`

```
long result;
```

- Wessen Rückgabewert kann result zugewiesene werden (3)
  Die Methode soll 2 Strings und 1 int entgegennehmen.

A. `Short myMethod1(String str1, int str2, String str3)`

B. `Int myMethod2(String val1, int val2, String val3)`

C. `Byte myMethod3(String str1, str2, int a)`

D. `Float myMethod4(String val1, val2, int val3)`

E. `Long myMethod5(int str2, String str3, String str1)`

F. `Long myMethod6(String... val1, int val2)`

G. `Short myMethod7(int val1, String... val2)`

```
class SwJava {
    public static void main(String args[]) {
        String[] shapes = {"Circle", "Square", "Triangle"};
        switch (shapes) {
            case "Square": System.out.println("Circle"); break;
            case "Triangle": System.out.println("Square"); break;
            case "Circle": System.out.println("Triangle"); break;
        }
    }
}
```

- Wähle die korrekte Aussage (1)

A. The code prints Circle.

B. The code prints Square.

C. The code prints Triangle.

D. The code prints
   Circle
   Square
   Triangle

E. The code prints
   Triangle
   Circle
   Square

F. The code fails to compile.

- Was davon kompiliert? (3)

A.
```
int eArr1[] = {10, 23, 10, 2};
```

B.
```
int[] eArr2 = new int[10];
```

C.
```
int[] eArr3 = new int[] {};
```

D.
```
int[] eArr4 = new int[10] {};
```

E.
```
int eArr5[] = new int[2] {10, 20};
```

- Welche Methode liefert einen zusammengesetzten String? (2)

A.
```
public String add(String 1, String 2) {
    return str1 + str2;
}
```

B.
```
private String add(String s1, String s2) {
    return s1.concat(s2);
}
```

C.
```
protected String add(String value1, String value2) {
    return value2.append(value2);
}
```

D.
```
String subtract(String first, String second) {
    return first.concat(second.substring(0));
}
```

```
int ctr = 10;
char[] arrC1 = new char[]{'P','a','u','l'};
char[] arrC2 = {'H','a','r','r','y'};
// CODE
System.out.println(ctr);
```

- Was könnte eingesetzt werden, um als Ausgabe 14 zu bekommen? (2)

A.
```
for (char c1 : arrC1) {
      for (char c2 : arrC2) {
          if (c2 == 'a') break;
              ++ctr;
      }
  }
```

B.
```
for (char c1 : arrC1)
      for (char c2 : arrC2) {
          if (c2 == 'a') break;
              ++ctr;
      }
```

C.
```
for (char c1 : arrC1)
      for (char c2 : arrC2)
          if (c2 == 'a') break;
              ++ctr;
```

D.
```
for (char c1 : arrC1) {
      for (char c2 : arrC2) {
          if (c2 == 'a') continue;
          ++ctr;
      }
  }
```

```
public class TestClass{
   public static void main(String args[ ] ){
      StringBuilder sb = new StringBuilder("12345678");
      sb.setLength(5);
      sb.setLength(10);
      System.out.println(sb.length());
   }
}
```

- What will be the result of attempting to compile and run the following program?
  (1)

A. It will print 5.

B. It will print 10.

C. It will print 8.

D. Compilation error.

E. None of the above.

```
final class Home {
    String name;
    int rooms;
    // CONSTRUCTOR
}
```

- Welche Optionen definieren ein zulässiges Überladen des Konstruktors (3)

A. `Home() {}`

B. `Float Home() {}`

C. `protected Home(int rooms) {}`

D. `final Home() {}`

E. `private Home(long name) {}`

F. `float Home(int rooms, String name) {}`

G. `static Home() {}`

```
for (int ctr = 2; ctr <= 30; ++ctr) {
    if (ctr % 7 != 0)
        //CODE
    if (ctr % 14 == 0)
        System.out.println(ctr);
}
```

- Welche CODE kann an markierter Position eingesetzt werden um als Ausgabe 14 und 28 zu bekommen? (1)

A. `continue;`

B. `exit;`

C. `break;`

D. `end;`

```
import java.util.function.Predicate;
public class MyCalendar {
    public static void main(String arguments[]) {
        Season season1 = new Season();
        season1.name = "Spring";

        Season season2 = new Season();
        season2.name = "Autumn";

        Predicate<String> aSeason = (s) -> s == "Summer" ? season1.name : season2.name;
        season1 = season2;
        System.out.println(season1.name);
        System.out.println(season2.name);
        System.out.println(aSeason.test(new String("Summer")));
    }
}
class Season {
    String name;
}
```

- Was ist die Ausgabe? (1)

A.
```
String
Autumn
false
```

B.
```
Spring
String
false
```

C.
```
Autumn
Autumn
false
```

D.
```
Autumn
String
true
```

E.
```
Compilation error
```

```
public class If2 {
    public static void main(String args[]) {
        int a = 10; int b = 20; boolean c = false;
        if (b > a) if (++a == 10) if (c!=true)
System.out.println(1);
        else System.out.println(2); else
System.out.println(3);
    }
}
```

- Was ist die Ausgabe? (1)

A. 1

B. 2

C. 3

D. No output

```
class Shoe {}
class Boot extends Shoe {}
class ShoeFactory {
    ShoeFactory(Boot val) {
        System.out.println("boot");
    }
    ShoeFactory(Shoe val) {
        System.out.println("shoe");
    }
}
```

- Was trifft für die obige Code zu? (1)

A. The class ShoeFactory has a total of two overloaded constructors.

B. The class ShoeFactory has three overloaded constructors, two user-defined constructors, and one default constructor.

C. The class ShoeFactory will fail to compile.

D. The addition of the following constructor will increment the number of constructors of the class ShoeFactory to 3:

```
private ShoeFactory (Shoe arg) {}
```

```
class ColorPencil {
    String color;
    ColorPencil(String color) {
        // CODE
    }
}
class TestColor {
    ColorPencil myPencil = new ColorPencil("RED");
}
```

- Welche Option belegt die Instanz-Variable color der Referenz-Variable myPencil mit einem String „RED“? (1)

A. `this.color = color;`

B. `color = color;`

C. `color = RED;`

D. `this.color = RED;`

```
class EJavaCourse {
    String courseName = "Java";
}
class University {
    public static void main(String args[]) {
        EJavaCourse courses[] = { new EJavaCourse(), new EJavaCourse() };
        courses[0].courseName = "OCA";
        for (EJavaCourse c : courses) c = new EJavaCourse();
        for (EJavaCourse c : courses) System.out.println(c.courseName);
    }
}
```

- Was ist die Ausgabe? (1)

A.
```
Java
Java
```

B.
```
OCA
Java
```

C.
```
OCA
OCA
```

D. None of the above

```
class MyExam {
    void question() {
        try {
            question();
        } catch (StackOverflowError e) {
            System.out.println("caught");
        }
    }
    public static void main(String args[]) {
        new MyExam().question();
    }
}
```

- Was trifft auf die obige Code zu? (3)

A. The code will print caught.

B. The code won't print caught.

C. The code would print caught if StackOverflowError were a runtime exception.

D. The code would print caught if StackOverflowError were a checked exception.

E. The code would print caught if question() throws the exception NullPointer-Exception.

```
public class Student {
    private String fName;
    private String lName;
    public Student(String first, String last) {
        fName = first; lName = last;
    }
    public String getName() { return fName + lName; }
}

// SPÄTERE ÄNDERUNG
public String getName() {
    return fName + " " + lName;
}
```

- Was sind die Auswirkungen der Änderung? (2)

A. The classes that were using the class Student will fail to compile.

B. The classes that were using the class Student will work without any compilation issues.

C. The class Student is an example of a well-encapsulated class.

D. The class Student exposes its instance variable outside the class.

```
class ColorPack {
    int shadeCount = 12;
    static int getShadeCount() {
        return shadeCount;
    }
}

class Artist {
    public static void main(String args[]) {
        ColorPack pack1 = new ColorPack();
        System.out.println(pack1.getShadeCount());
    }
}
```

- Was ist die Ausgabe? (1)

A. 10

B. 12

C. No output

D. Compilation error

```
class Laptop {
    String memory = "1 GB";
}

class Workshop {
    public static void main(String args[]) {
        Laptop life = new Laptop();
        repair(life);
        System.out.println(life.memory);
    }

    public static void repair(Laptop laptop) {
        laptop.memory = "2 GB";
    }
}
```

- Was ist die Ausgabe? (1)

A. 1 GB

B. 2 GB

C. Compilation error

D. Runtime exception

```
public class Application {
    public static void main(String... args) {
        double price = 10;
        String model;
        if (price > 10)
            model = "Smartphone";
        else if (price <= 10)
            model = "landline";
        System.out.println(model);
    }
}
```

- Was ist die Ausgabe? (1)

A. landline

B. Smartphone

C. No output

D. Compilation error

```
class MyNumString {
    public static void main(String args[]) {
        String numStr = "123456789";
        System.out.println(numStr.substring(numStr.indexOf("2"),
        numStr.indexOf("0")).concat("0"));
    }
}
```

- Was ist die Ausgabe? (1)

A. 234567890

B. 34567890

C. 234456789

D. 3456789

E. Compilation error

F. Runtime exception

```
class Artist {
    Artist assistant;
}
class Studio {
    public static void main(String... args) {
        Artist a1 = new Artist();
        Artist a2 = new Artist();
        a2.assistant = a1;
        a2 = null; // Line 1
    }
     // Line 2
}
```

- Wähle die korrekten Aussagen (2)

A. At least two objects are garbage collected on line 1.

B. At least one object is garbage collected on line 1.

C. No objects are garbage collected on line 1.

D. The number of objects that are garbage collected on line 1 is unknown.

E. At least two objects are eligible for garbage collection on line 2.

```
class Book {
    String ISBN;
    Book(String val) {
        ISBN = val;
    }
}
class TestEquals {
    public static void main(String... args) {
        Book b1 = new Book("1234-4657");
        Book b2 = new Book("1234-4657");
        System.out.print(b1.equals(b2) +":");
        System.out.print(b1 == b2);
    }
}
```

- Was ist die Ausgabe? (1)

A. true:false

B. true:true

C. false:true

D. false:false

E. Compilation error – there is no equals method in the class Book.

F. Runtime exception.

- Wähle die korrekten Aussagen (2)

A. `StringBuilder sb1 = new StringBuilder()` will create a StringBuilder object with no characters but with an initial capacity to store 16 characters.

B. `StringBuilder sb1 = new StringBuilder(5*10)` will create a StringBuilder object with a value of 50.

C. Unlike the class String, the concat method in StringBuilder modifies the value of a StringBuilder object.

D. The insert method can be used to insert a character, number, or String at the start or end or a specified position of a StringBuilder.

```
interface Jump {}
class Animal implements Jump {}
```

- Wähler die korrekten Array Deklarationen und Initialisierungen aus (3)

A. `Jump eJump1[] = {null, new Animal()};`

B. `Jump[] eJump2 = new Animal()[22];`

C. `Jump[] eJump3 = new Jump[10];`

D. `Jump[] eJump4 = new Animal[87];`

E. `Jump[] eJump5 = new Jump()[12];`

```
class EIf {
    public static void main(String args[]) {
        bool boolean = false;
        do {
            if (boolean = true)
                System.out.println("true");
            else
                System.out.println("false");
        }
        while(3.3 + 4.7 > 8);    }
}
```

- Was ist die Ausgabe? (1)

A. The class will print true.

B. The class will print false.

C. The class will print true if the if condition is changed to boolean == true.

D. The class will print false if the if condition is changed to boolean != true.

E. The class won't compile.

F. Runtime exception.

```
class Whale {
    public static void main(String args[]) {
        boolean hungry = false;
        while (hungry=true) {
            ++Fish.count;
        }
        System.out.println(Fish.count);
    }
}
class Fish {
    static byte count;
}
```

- Wähle die korrekten Aussagen (2)

A. The code doesn’t compile.

B. The code doesn’t print a value.

C. The code prints 0.

D. Changing ++Fish.count to Fish.count++ will give the same results.

```
class Phones {
    public static void main(String args[]) {
        String phones[]= {"BlackBerry", "Android", "iPhone"};
        for (String phone : phones)
          /* CODE */
    }
}
```

- Was muss als CODE eingesetzt werden, damit der Telefonname und Position an dem es im Array steht ausgegeben wird? (1)

A. `System.out.println(phones.count + ":" + phone);`

B. `System.out.println(phones.counter + ":" + phone);`

C. `System.out.println(phones.getPosition() + ":" + phone);`

D. `System.out.println(phones.getCtr() + ":" + phone);`

E. `System.out.println(phones.getCount() + ":" + phone);`

F. `System.out.println(phones.pos + ":" + phone);`

G. None of the above

```
Byte b1 = (byte)100;                          // 1
Integer i1 = (int)200;                        // 2
Long l1 = (long)300;                          // 3
Float f1 = (float)b1 + (int)l1;               // 4
String s1 = 300;                              // 5
if (s1 == (b1 + i1))                          // 6
    s1 = (String)500;                         // 7
else                                          // 8
    f1 = (int)100;                            // 9
System.out.println(s1 + ":" + f1);            // 10
```

- Was ist die Ausgabe? (1)

A. Code fails compilation at line numbers 1, 3, 4, 7.

B. Code fails compilation at line numbers 6, 7.

C. Code fails compilation at line numbers 7, 9.

D. Code fails compilation at line numbers 4, 5, 6, 7, 9.

E. No compilation error—outputs 500:300.

F. No compilation error—outputs 300:100.

G. Runtime exception.

```
class Book {
    String ISBN;
    Book(String val) {
        ISBN = val;
    }
    public boolean equals(Object b) {
        if (b instanceof Book) {
            return ((Book)b).ISBN.equals(ISBN);
        }
        else
            return false;
    }
}

class TestEquals {
    public static void main(String args[]) {
        Book b1 = new Book("1234-4657");
        Book b2 = new Book("1234-4657");
        LocalDate release = null;
        release = b1.equals(b2) ? b1 == b2? LocalDate.of(2050,12,12):
        LocalDate.parse("2072-02-01"):LocalDate.parse("9999-09-09");
        System.out.print(release);
    }
}
```

- Was ist die Ausgabe? (1)

    A. `2050-12-12`

    B. `2072-02-01`

    C. `9999-09-09`

    D. `Compilation error`

    E. `Runtime exception`

```
class Raft {
    public String rideWave() { return null; }
    //CODE
}
```

- Was kann als CODE eingesetzt werden um die Methode rideWave zu überladen? (1)

A. `public String[] rideWave() { return null; }`

B. `protected void riceWave(int a) {}`

C. `private void rideWave(int value, String value2) {}`

D.
```
default StringBuilder rideWave (StringBuffer a)
  { return null; }
```

```
int num[] = {10, 15, 2, 17};
int sum = 0;
for (int number : num) {
    //CODE
    sum += number;
}
```

- Was kann als CODE eingesetzt werden um die Zahlen im Array num zu summieren und das Ergebnis in sum abzulegen ? (1)

A.
```
if (number % 2 == 0)
    continue;
```

B.
```
if (number % 2 == 0)
    break;
```

C.
```
if (number % 2 != 0)
    continue;
```

D.
```
if (number % 2 != 0)
    break;
```

```
class Op {
    public static void main(String... args) {
        int a = 0;
        int b = 100;
        Predicate<Integer> compare = (var) -> var++ == 10;
        if (!b++ > 100 && compare.test(a)) {
            System.out.println(a+b);
        }
    }
}
```

- Was ist die Ausgabe? (1)

A. 100

B. 101

C. 102

D. Code fails to compile.

- Create a well-encapsulated class Pencil with one instance variable model. The value of model should be accessible and modifiable outside Pencil. (1)

A.
```
class Pencil {
      public String model;
  }
```

B.
```
class Pencil {
      public String model;
      public String getModel() { return model; }
      public void setModel(String val) { model = val; }
  }
```

C.
```
class Pencil {
      private String model;
      public String getModel() { return model; }
      public void setModel(String val) { model = val; }
  }
```

D.
```
class Pencil {
      public String model;
      private String getModel() { return model; }
      private void setModel(String val) { model = val; }
  }
```

```
class Diary {
    int pageCount = 100;
    int getPageCount() {
        return pageCount;
    }
    void setPageCount(int val) {
        pageCount = val;
    }
}
class ClassRoom {
    public static void main(String args[]) {
        System.out.println(new Diary().getPageCount());
        new Diary().setPageCount(200);
        System.out.println(new Diary().getPageCount());
    }
}
```

- Was ist die Ausgabe? (1)

A. 100
   200

B. 100
   100

C. 200
   200

D. Code fails to compile.

```
class Shopping {
    public static void main(String args[]) {
        boolean bankrupt = true;
        do System.out.println("enjoying shopping");
bankrupt = false;
        while (!bankrupt);
    }
}
```

- Was ist die Ausgabe? (1)

A. The code prints enjoying shopping once.

B. The code prints enjoying shopping twice.

C. The code prints enjoying shopping in an infinite loop.

D. The code fails to compile.

- Was sind valide Definitionen von multidimensionalen Arrays? (4)

A.
```
String ejg1[][] = new String[1][2];
```

B.
```
String ejg2[][] = new String[][] { {}, {} };
```

C.
```
String ejg3[][] = new String[2][2];
```

D.
```
String ejg4[][] = new String[][]{{null},new String[]
  {"a","b","c"}, {new String()}};
```

E.
```
String ejg5[][] = new String[][2];
```

F.
```
String ejg6[][] = new String[][]{"A", "B"};
```

G.
```
String ejg7[][] = new String[]{{"A"}, {"B"}};
```

```
class Laptop {
    String memory = "1GB";
}
class Workshop {
    public static void main(String args[]) {
        Laptop life = new Laptop();
        repair(life);
        System.out.println(life.memory);
    }
    public static void repair(Laptop laptop) {
        laptop = new Laptop();
        laptop.memory = "2GB";
    }
}
```

- Was ist die Ausgabe? (1)

A. 1 GB

B. 2 GB

C. Compilation error

D. Runtime exception

```
interface Roamable{}
class Phone {}
class Tablet extends Phone implements Roamable {
    //CODE
}
```

- Welche CODE ermöglicht es einer Referenz-Variable vom Typ Roamable auf ein Phone Objekt zu verweisen? (1)

A. `Roamable var = new Phone();`

B. `Roamable var = (Roamable)Phone();`

C. `Roamable var = (Roamable)new Phone();`

D. Because the interface Roamable and the class Phone are unrelated, a reference variable of type Roamable can't refer to an object of the class Phone.

```
class Paper {
    Paper() {
        this(10);
        System.out.println("Paper:0");
    }
    Paper(int a) { System.out.println("Paper:1"); }
}
class PostIt extends Paper {}
class TestPostIt {
    public static void main(String[] args) {
        Paper paper = new PostIt();
    }
}
```

- Was ist die Ausgabe? (1)

A.
```
Paper:1
```

B.
```
Paper:0
```

C.
```
Paper:0
Paper:1
```

D.
```
Paper:1
Paper:0
```

```
interface Jumpable {
    int height = 1;
    default void worldRecord() {
        System.out.print(height);
    }
}
interface Moveable {
    int height = 2;
    static void worldRecord() {
        System.out.print(height);
    }
}
class Chair implements Jumpable, Moveable {
    int height = 3;
    Chair() {
        worldRecord();
     }
    public static void main(String args[]) {
        Jumpable j = new Chair();
        Moveable m = new Chair();
        Chair c = new Chair();
    }
}
```

- Was ist die Ausgabe? (1)

    A. 111

    B. 123

    C. 333

    D. 222

    E. Compilation error

    F. Runtime exception

```
class Animal{ float age; }
class Lion extends Animal { int claws;}
class Jungle {
    public static void main(String args[]) {
        Animal animal = new Lion();
        /* CODE
        System.out.println(1);
    }
}
```

- Mit welcher CODE kann die Klasse Jungle testen, ob die Referenzvariable Animal auf ein Objekt der Klasse Lion verweist und dann eine 1 ausgibt? (1)

A. `if (animal instanceof Lion)`

B. `if (animal instanceOf Lion)`

C. `if (animal == Lion)`

D. `if (animal = Lion)`

```
class Person {
    Person(String value) {}
}
class Employee extends Person {}
class Test {
    public static void main(String args[]) {
        Employee e = new Employee();
    }
}
```

- Finde den Grund für den Compilerfehler? (2)

A. The class Person fails to compile.

B. The class Employee fails to compile.

C. The default constructor can call only a no-argument constructor of a base class.

D. The code that creates the object of the class Employee in the class Test did not pass a String value to the constructor of the class Employee.

```
class Book {
    private int pages = 100;
}
class Magazine extends Book {
    private int interviews = 2;
    private int totalPages() { /* INSERT CODE HERE */ }
    public static void main(String[] args) {
        System.out.println(new Magazine().totalPages());
    }
}
```

- Welche CODE sorgt dafür, dass 110 ausgegeben wird? (1)

A. `return super.pages + this.interviews*5;`

B. `return this.pages + this.interviews*5;`

C. `return super.pages + interviews*5;`

D. `return pages + this.interviews*5;`

E. None of the above

```
class NoInkException extends Exception {}
class Pen{
    void write(String val) throws NoInkException {
        int c = (10 - 7)/ (8 - 2 - 6);
    }
    void article() {
        // CODE
    }
}
```

- Welche CODE definiert eine valide Verwendung der Methode write in der Methode article? (2)

A.
```
try {
      new Pen().write("story");
  } catch (NoInkException e) {}
```

B.
```
try {
      new Pen().write("story");
  } finally {}
```

C.
```
try {
      write("story");
  } catch (Exception e) {}
```

D.
```
try {
      new Pen().write("story");
  } catch (RuntimeException e) {}
```

```
class EMyMethods {
    static String name = "m1";
    void riverRafting() {
        String name = "m2";
        if (8 > 2) {
            String name = "m3";
            System.out.println(name);
        }
    }
    public static void main(String[] args) {
        EMyMethods m1 = new EMyMethods();
        m1.riverRafting();
    }
}
```

- Was ist die Ausgabe? (1)

A. m1

B. m2

C. m3

D. The code fails to compile.

```
class EBowl {
    public static void main(String args[]) {
        String eFood = "Corn";
        System.out.println(eFood);
        mix(eFood);
        System.out.println(eFood);
    }
    static void mix(String foodIn) {
        foodIn.concat("A");
        foodIn.replace('C', 'B');
    }
}
```

- Was ist die Ausgabe? (1)

A.
```
Corn
BornA
```

B.
```
Corn
CornA
```

C.
```
Corn
Born
```

D.
```
Corn
Corn
```

```
class Phone {
    static void call() {
        System.out.println("Call-Phone");
    }
}
class SmartPhone extends Phone{
    static void call() {
        System.out.println("Call-SmartPhone");
    }
}
class TestPhones {
    public static void main(String... args) {
        Phone phone = new Phone();
        Phone smartPhone = new SmartPhone();
        phone.call();
        smartPhone.call();
    }
}
```

- Was ist die Ausgabe? (1)

A.
```
Call-Phone
Call-Phone
```

B.
```
Call-Phone
Call-SmartPhone
```

C.
```
Call-Phone
null
```

D.
```
null
Call-SmartPhone
```

```
class Person {}
class Father extends Person {
    public void dance() throws ClassCastException {}
}
class Home {
    public static void main(String args[]) {
        Person p = new Person();
        try {
            ((Father)p).dance();
        }
        //CODE
    }
}
```

- Welche CODE kann eingesetzt werden, damit das Programm erfolgreich kompiliert? (3)

A.
```
catch (NullPointerException e) {}
catch (ClassCastException e) {}
catch (Exception e) {}
catch (Throwable t) {}
```

B.
```
catch (ClassCastException e) {}
catch (NullPointerException e) {}
catch (Exception e) {}
catch (Throwable t) {}
```

C.
```
catch (ClassCastException e) {}
catch (Exception e) {}
catch (NullPointerException e) {}
catch (Throwable t) {}
```

D.
```
catch (Throwable t) {}
catch (Exception e) {}
catch (ClassCastException e) {}
catch (NullPointerException e) {}
```

E.
```
finally {}
```

```
class Course {
    int enrollments;
}
class TestEJavaCourse {
    public static void main(String args[]) {
        Course c1 = new Course();
        Course c2 = new Course();
        c1.enrollments = 100;
        c2.enrollments = 200;
        System.out.println(c1.enrollments + c2.enrollments);
    }
}
```

- Die Ausgabe der Code ist aktuell 300. Was ändert sich, wenn die Variable enrollments statisch wird? (1)

A. No change in output. TestEJavaCourse prints 300.

B. Change in output. TestEJavaCourse prints 200.

C. Change in output. TestEJavaCourse prints 400.

D. The class TestEJavaCourse fails to compile.

```
import java.util.*;
class Person {}
class Emp extends Person {}
class TestArrayList {
    public static void main(String[] args) {
        ArrayList<Object> list = new ArrayList<>();
        list.add(new String("1234"));                  //LINE1
        list.add(new Person());                        //LINE2
        list.add(new Emp());                           //LINE3
        list.add(new String[]{"abcd", "xyz"});         //LINE4
        list.add(LocalDate.now().plus(1));             //LINE5
    }
}
```

- Wähle die korrekte Aussage (1)

A. The code on line 1 won't compile.

B. The code on line 2 won't compile.

C. The code on line 3 won't compile.

D. The code on line 4 won't compile.

E. The code on line 5 won't compile.

F. None of the above.

G. All the options from (a) through (e).

```
interface Movable {
    default int distance() {
        return 10;
    }
}

interface Jumpable {
    default int distance() {
        return 10;
    }
}
```

- Welche option implementiert korrekt Movable und Jumpable? (1)

A.
```
class Person implements Movable, Jumpable {}
```

C.
```
class Person implements Movable, Jumpable {
      default int distance() {
          return 10;
      }
  }
```

D.
```
class Person implements Movable, Jumpable {
      public int distance() {
          return 10;
      }
  }
```

E.
```
class Person implements Movable, Jumpable {
      public long distance() {
          return 10;
      }
  }
```

G.
```
class Person implements Movable, Jumpable {
      int distance() {
          return 10;
      }
  }
```