

# Simulado para OCJP Java Programmer - 1Z0-808

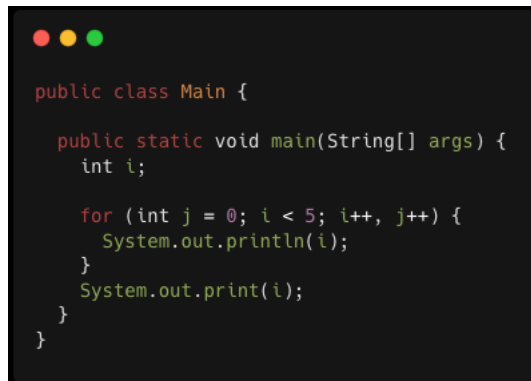
56 Questões.

Tempo de prova 120 minutos.

65% de acertos para aprovação.

Autor: Silvano Malfatti

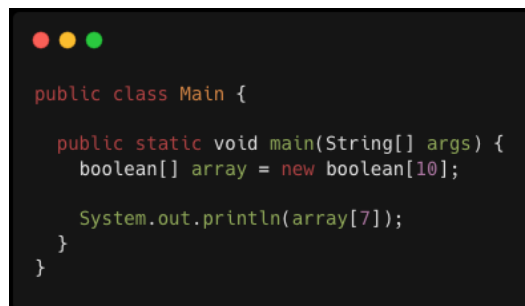
**1) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**



```
public class Main {  
  
    public static void main(String[] args) {  
        int i;  
  
        for (int j = 0; i < 5; i++, j++) {  
            System.out.println(i);  
        }  
        System.out.print(i);  
    }  
}
```

- a) Erro de compilação
- b) Compila e roda, imprimindo de 0 a até 4
- c) Compila e roda, imprimindo de 0 a até 5
- d) Compila e roda, mas dispara uma exceção em tempo de execução.

**2) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**



```
public class Main {  
  
    public static void main(String[] args) {  
        boolean[] array = new boolean[10];  
  
        System.out.println(array[7]);  
    }  
}
```

- a) Erro de compilação
- b) Compila e roda, imprimindo null
- c) Compila e roda, imprimindo true
- d) Compila e roda, imprimindo false
- e) Compila e roda, mas dispara uma NullPointerException

**3) Escolha duas palavras reservadas que não fazem parte da linguagem Java.**

- a) instanceof
- b) native
- c) Extends
- d) short
- e) include

**4) Analise o código a seguir e escreva quantos objetos estarão elegíveis para o coletor de lixo após a execução da linha indicada.**

```
public class Main {  
    public static void main(String[] args) {  
        String name = "";  
        for (int position = 0; position < 10; position++) {  
            name = getName(position);  
        } // <-----  
    }  
  
    static String getName(int position) {  
        return new String("Name " + position);  
    }  
}
```

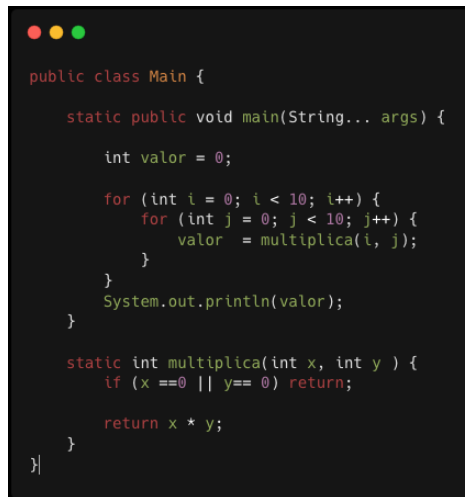
- a) Apenas 1
- b) 10 objetos poderão ser coletados
- c) 11 objetos poderão ser coletados
- d) 2 objetos poderão ser coletados
- e) Nenhum objeto estará disponível para o coletor de lixo

**5) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**

```
public class Main {  
    public static void main(String[] args) {  
        for (int position = 0; position < 10; position++) {  
            if (position % 2 == 0) continue;  
            System.out.println(position);  
        }  
    }  
}
```

- a) O código não compila
- b) O código compila, roda e imprime todos os números de 0 a 9
- c) O código compila e imprime os valores 0, 2, 4, 8
- d) O código compila e imprime os valores 1, 3, 5, 7, 9
- e) O código compila e dispara uma exceção em tempo de execução

**6) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**



```
public class Main {  
  
    static public void main(String... args) {  
  
        int valor = 0;  
  
        for (int i = 0; i < 10; i++) {  
            for (int j = 0; j < 10; j++) {  
                valor = multiplica(i, j);  
            }  
        }  
        System.out.println(valor);  
    }  
  
    static int multiplica(int x, int y) {  
        if (x == 0 || y == 0) return;  
  
        return x * y;  
    }  
}
```

- a) O código não compila
- b) O código compila, roda e imprime o valor zero.
- c) O código compila, roda e imprime apenas o número 81
- d) O código compila, roda e imprime o valor 100

7) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
abstract class Person {
    String name = "superclass";

    @Override
    String toString() {
        return name;
    }
}

class Programmer extends Person {
    String language = "Java";

    @Override
    String toString() {
        return language;
    }
}

public class Main {

    public static void main(String args[]) {

        Person reference = (Person) new Programmer();

        System.out.println(reference);
    }
}
```

- a) O código não compila
- b) O código compila, roda e imprime "superclass"
- c) O código compila, roda e imprime "Java"
- d) O código compila, roda e dispara uma exceção durante sua execução.

8) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
abstract class Person {
    String name = "superclass";

    String getName() {
        return this.name;
    }
}

class Programmer extends Person {
    String name = "Programmer";

    @Override
    String getName() {
        return this.name;
    }
}

public class Main {

    public static void main(String args[]) {

        Person reference = (Programmer)(Person) new Programmer();

        System.out.println(reference.getName());
    }
}
```

- a) O código não compila
- b) O código compila, roda e imprime "superclass"
- c) O código compila, roda e imprime "Programmer"
- d) O código compila, roda e dispara uma exceção durante sua execução.

9) O código a seguir apresenta um erro de compilação. Qual das alternativas a seguir corrigiria o problema e faria o código compilar com sucesso:

```
class DataBase {  
    public boolean openDB(String name) {  
        open(name);  
        return true;  
    }  
  
    private void open(String name) throws Exception {  
        if (name.isEmpty()) {  
            throw new Exception();  
        }  
  
        System.out.println("BD openned");  
    }  
}  
  
public class Main {  
    public static void main(String args[]) {  
        DataBase db = new DataBase();  
        db.openDB("DBName");  
    }  
}
```

a) `public static void main(String args[]) throws Exception {`

b) `try {  
 open(name); } catch (Exception e) {}`

c) `public boolean openDB(String name) throws Exception{`

d) `try {  
 DataBase db = new DataBase();  
 db.openDB("DBName");  
} catch (Exception e) {}`

**10) Escolha a opção adequada ao tentar compilar e rodar o código a seguir (considere que cada classe esteja em seus respectivos pacotes):**

```
package generic;

public abstract class Vehicle {
    public Vehicle(){}
    public abstract String getType();
}

package core;

import generic.*;

public class Airplane extends Vehicle {
    public String getType() {
        return "Airplane";
    }
}

public class Main {
    public static void main(String args) {
        Vehicle reference = (Vehicle) new Airplane();
        System.out.println(reference.getType());
    }
}
```

- a) O código não compila
- b) O código compila, roda e imprime "Airplane"
- c) O código compila, roda e imprime o endereço do objeto
- d) O código compila, roda e dispara uma exceção durante sua execução.

**11) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**

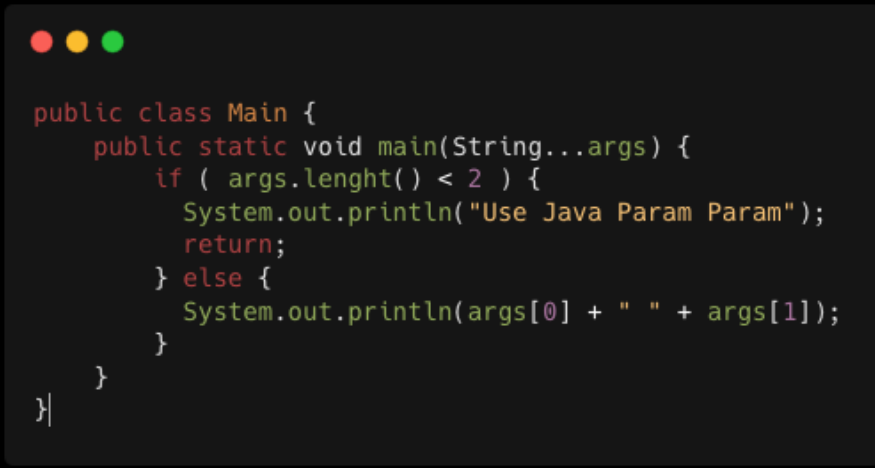
```
public class Main {
    public static void main(String args) {
        int value = 0 ;

        if (value == 0) {
            return;
        }

        value = value / value;
        System.out.println(value);
    }
}
```

- a) O código não compila
- b) O código compila, roda e imprime o valor zero.
- c) O código compila, roda e não imprime nada
- d) O código compila, roda e dispara uma exceção em tempo de execução.

12) Escolha a opção adequada ao tentar compilar e rodar o código a seguir utilizando o comando `c:\java Main parametro0 parametro1`



```
public class Main {  
    public static void main(String...args) {  
        if ( args.lenght() < 2 ) {  
            System.out.println("Use Java Param Param");  
            return;  
        } else {  
            System.out.println(args[0] + " " + args[1]);  
        }  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime Use Java Param Param 2
- c) O código compila e imprime Parametro0 Parametro1
- d) O código compila e imprime Main Parametro1

13) Qual o tipo de exceção a seguir é considerada uma **Unchecked Exception**

- a) IOException
- b) NullPointerException
- c) FileNotFoundException
- d) OutOfMemoryError

14) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
public class Main {
    public static void main(String args[]) {

        int value = 0;
        int number = 12;

        try {
            System.out.println("Step 1");
            int result = number / value;
            System.out.println("Step 2");
        }
        catch (ClassCastException e) {
            System.out.println("Step 3");
        }
        catch (ArithmeticException e) {
            System.out.println("Step 4");
        }
        finally {
            System.out.println("Step 5");
        }
        System.out.println("Step 6");
    }
}
```

- a) O código não compila
- b) O código compila e imprime os passos 1, 2, 4, 5, 6
- c) O código compila e imprime os passos 1, 4, 5, 6
- d) O código compila, roda e termina sua execução devido a uma exceção não tratada

15) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
class Person {
    public String toString() {
        return "Just a person";
    }
}

class Programmer extends Person {
    int numberOfHours = 10;

    Programmer(int hours) {
        super();
        numberOfHours = hours;
    }

    public String toString() {
        return "A programmer";
    }

    public int getSalary() {
        return numberOfHours * 100;
    }
}

public class Main {
    public static void main(String args[]) {

        Person p = new Programmer(40);

        System.out.println(p.getSalary());
    }
}
```

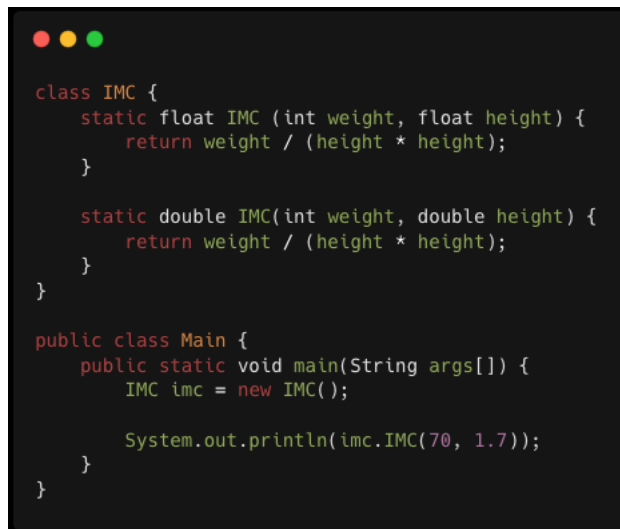


- a) O código não compila
- b) O código compila, roda e dispara uma exceção
- c) O código compila, roda e imprime 1000
- d) O código compila, roda e imprime zero

**16) Quais as atribuições a seguir geram um erro de compilação (selecione duas):**

- a) `int number = 0x456;`
- b) `char letter = "c";`
- c) `float height = 1.7;`
- d) `short negative = -0456;`
- e) `byte intruction = -128;`

**17) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**



```
class IMC {  
    static float IMC (int weight, float height) {  
        return weight / (height * height);  
    }  
  
    static double IMC(int weight, double height) {  
        return weight / (height * height);  
    }  
}  
  
public class Main {  
    public static void main(String args[]) {  
        IMC imc = new IMC();  
  
        System.out.println(imc.IMC(70, 1.7));  
    }  
}
```

- a) O código não compila
- b) O código compila e dispara uma exceção durante sua execução
- c) O código compila e invoca o método `float IMC(int, float)`
- d) O código compila e invoca o método `double IMC(int, double)`

18) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

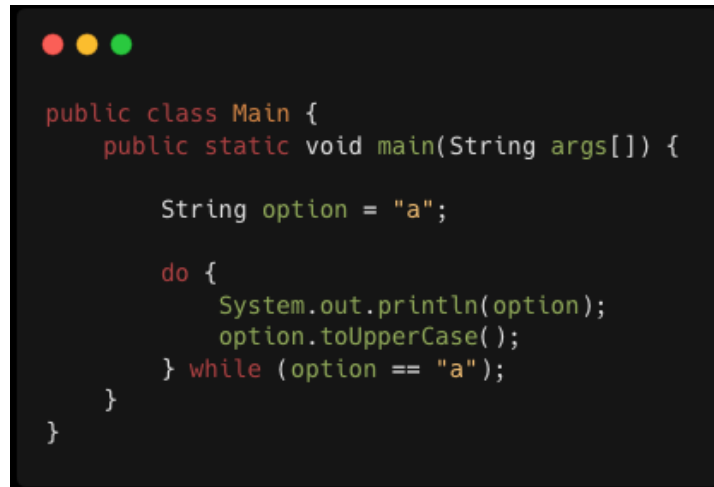
```
class Person {  
    String name;  
    int age;  
}  
  
public class Main {  
    public static void main(String args[]) {  
  
        Person p = new Person();  
        p.name = "Luis";  
        p.age = 30;  
  
        reset(p.name, p.age);  
  
        System.out.println(p.name);  
        System.out.println(p.age);  
    }  
  
    static void reset (String name, int age) {  
        name = "Default";  
        age = 0;  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime Luis, 30
- c) O código compila e imprime Default, 30
- d) O código compila e imprime Default, 0

19) Qual das declarações a seguir não está correta (pelo menos duas):

- a) `Int[] array = new int[10]();`
- b) `Int array[] = {1,2,3,4,5};`
- c) `Int[] array[] = new int[][];`
- d) `Int array[][] = {{1,2}, {1,7}};`
- e) `Int[][][] array = new int[10][][];`

**20) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:**



```
public class Main {  
    public static void main(String args[]) {  
  
        String option = "a";  
  
        do {  
            System.out.println(option);  
            option.toUpperCase();  
        } while (option == "a");  
    }  
}
```

- a) O código não compila
- b) O código compila, roda mas gera uma exceção do tipo StackOverflow()
- c) O código compila, roda e imprime a letra “a” apenas uma vez
- d) O código compila, roda e entra em loop infinito

**21) Quais das expressões Lambda a seguir estão corretas?**

- a) () -> “Hello World”
- b) (int x) -> x \* x
- c) (String s) -> { s.lenght(); }
- d) (String s, int x) -> s.substring(x);
- e) (int x, int y) -> { return x + y; }

**22) Qual das declarações a seguir não está correta para a assinatura de um método:**

- a) static int getIdade()
- b) protected void setName(String name)
- c) public final abstract ligarCarro()
- d) private Car()

23) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
interface Vehicle {  
    void turnOn();  
}  
  
class Beetle implements Vehicle {  
    void turnOn() {  
        System.out.println("Beetle on");  
    }  
}  
  
public class Main {  
    public static void main(String args[]) {  
  
        Vehicle beetle = new Beetle();  
        beetle.turnOn();  
    }  
}
```

- a) O código não compila
- b) O código compila, roda mas não imprime nada
- c) O código compila, roda e imprime Beetle on
- d) O código compila, roda mas dispara uma ClassCastException em tempo de execução

24) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
public class Main {  
    public static void main(String args[]) {  
  
        System.out.println((((4/3 == 1) == true) ? 1 : 0) == 0);  
    }  
}
```

- a) Imprime true
- b) Imprime false
- c) Imprime 0
- d) Imprime 1
- e) Não compila

25) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
public class Main {  
    public static void main(String args[]) {  
  
        String word1 = "home";  
        String word2 = word1.substring(0,4);  
  
        System.out.println(word1 == word2);  
        System.out.println(word1.equals(word2));  
    }  
}
```

- a) Não compila
- b) Imprime true, false
- c) Imprime true, true
- d) Imprime false, true
- e) Imprime false, false

26) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
interface Vehicle {}  
interface Engine {}  
interface Autobots extends Vehicle, Engine {}  
class Bot {}  
class Bumblebee extends Bot implements Autobots {}  
class OptimusPrime extends Bot implements Vehicle, Engine {}  
class Camaro extends Bumblebee {}  
  
public class Main {  
    public static void main(String args[]) {  
  
        Camaro camaro = (Camaro) new Bot();  
    }  
}
```

- a) Não compila na definição de classes e interfaces
- b) Não compila dentro do método Main
- c) Compila, roda e não imprime nada
- d) Compila, roda e dispara uma exceção

27) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
public class Main {  
    public static void main(String args[]) {  
  
        countdown: for (int counter = 10; counter >= 0; counter--) {  
            if (counter % 2 == 0) {  
                System.out.println(counter);  
                continue countdown;  
            }  
            else {  
                counter--;  
            }  
        }  
    }  
}
```

- a) Não compila
- b) Compila e imprime os números de 10 até 0
- c) Compila e não imprime nada
- d) Compila e imprime 10
- e) Compila, roda e entra em loop infinito

28) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
public class Main {  
    public static void main(String args[]) {  
  
        int value = 15;  
        int reference = value;  
  
        reference++;  
        value++;  
  
        int copy = reference;  
  
        copy--;  
  
        System.out.println(value + reference + copy);  
    }  
}
```

- a) Imprime 43
- b) Imprime 44
- c) Imprime 45
- d) Imprime 46
- e) Imprime 47
- f) Imprime 48
- g) Imprime 49

29) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
class Bot {
    void printVersion() {
        String version;

        if (!version.isEmpty()) {
            System.out.println(version);
        } else {
            System.out.println("No version");
        }
    }
}

public class Main {
    public static void main(String args[]) {
        Bot bot = new Bot();
        bot.printVersion();
    }
}
```

- a) Não compila
- b) Compila, mas dispara uma exceção ao executar
- c) Compila, roda e imprime No version
- d) Compila, roda e imprime null

30) Escolha a opção adequada ao tentar compilar e rodar o código a seguir:

```
abstract class Car {
    void printInfo() {
        System.out.println( getYear() );
    }

    int getYear() {
        return 0;
    }
}

abstract class OffRoadCar extends Car {
    int getYear() {
        return 1900;
    }
}

class Jeep extends OffRoadCar {
    int getYear() {
        return 1998;
    }
}

public class Main {
    public static void main(String args[]) {
        Jeep jeep = (Jeep) (OffRoadCar) new Jeep();
        jeep.printInfo();
    }
}
```

- a) Não compila
- b) Compila e imprime 1998
- c) Compila e imprime 0
- d) Compila e imprime 1900
- e) Compila, roda e dispara uma exception.

**31) Analise o código a seguir e escolha a opção adequada para sua compilação e execução. (Considere que todos os pacotes necessários foram importados)**

```
class TryException {  
    void openFile(String file) throws FileNotFoundException {  
        throw FileNotFoundException();  
    }  
  
    void getValue(int position) throws ArrayIndexOutOfBoundsException {  
        throw IndexOutOfBoundsException();  
    }  
}  
  
public class Main {  
    public static void main(String args[]) {  
  
        TryException exception = new TryException();  
        exception.getValue(10);  
  
    }  
}
```

- a) Não compila
- b) Compila e ao executar dispara uma exceção
- c) Compila e executa sem imprimir nada.
- d) Compila, executa e dispara uma exceção imprimindo informações sobre ela

**32) O que ocorrerá ao tentarmos compilar e executar o código a seguir:**

```
public class Main {  
    public static void main(String args[]) {  
  
        System.out.printf("%f %n", 23.9f);  
  
    }  
}
```



- a) O Código não compila
- b) O Código compila e exibe o valor 23
- c) O Código compila e exibe o valor 23.9
- d) O Código compila e exibe o valor 23.9000
- e) O Código compila e exibe o valor 23.90

**33) O que ocorrerá ao tentarmos compilar e executar o código a seguir:**

```
import java.time.LocalDateTime;

public class Main {
    public static void main(String args[]) {

        LocalDateTime today = LocalDateTime.of(2023,1,10,16,20);

        System.out.println(today.getMonth());
        System.out.println(today.getDayOfMonth());
    }
}
```

- a) O Código não compila
- b) O Código compila, roda e dispara uma exceção
- c) O Código compila, roda e imprime JANUARY 10
- d) O Código compila, roda e imprime 01 10

**34) Analise o código a seguir e assinale a alternativa correta:**

```
class PrimitiveTypes {

    int number;
    String name;
    double pi;

    void PrimitiveTypes() {

        number = 200;
        name = "Default";
        pi = 3.14;
    }
}

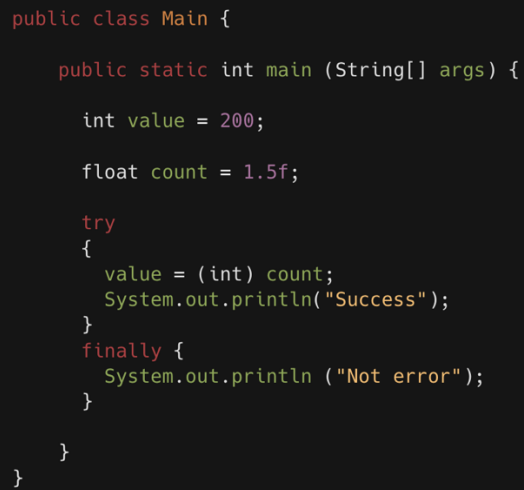
public class Main {
    public static void main(String args[]) {

        PrimitiveTypes types = new PrimitiveTypes();
        resetValue(types.pi);
        System.out.println(types.pi);
    }

    static void resetValue(double value) {
        value = 0;
    }
}
```

- a) O código não compila
- b) O código compila e imprime zero
- c) O código compila e imprime 3.14
- d) O código compila, roda mas não imprime nada

**35) Analise o código a seguir e assinale a alternativa correta:**



```
public class Main {  
    public static int main (String[] args) {  
        int value = 200;  
        float count = 1.5f;  
        try  
        {  
            value = (int) count;  
            System.out.println("Success");  
        }  
        finally {  
            System.out.println ("Not error");  
        }  
    }  
}
```

- a) O código não compila
- b) O código compila roda e exibe as mensagens Success Not error
- c) O código compila roda e exibe a mensagem Not error
- d) O código compila roda e dispara uma exceção

36) Analise o código a seguir e assinale a alternativa correta:

```
class Static {  
    public static int countRefs = 0;  
  
    Static() {  
        countRefs++;  
    }  
}  
  
public class Main {  
    public static void main (String[] args) {  
  
        new Static();  
        new Static();  
        new Static();  
        new Static();  
  
        System.out.println(Static.countRefs);  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime zero
- c) O código compila e imprime 1
- d) O código compila e imprime 4
- e) O código compila e imprime 5

37) Analise o código a seguir e assinale a alternativa correta:

```
class Car {  
    String name;  
    String model;  
    int year;  
}  
  
public class Main {  
    Car car = new Car();  
  
    void setCar(Car car) {  
        car = car;  
    }  
  
    Car getCar() {  
        return car;  
    }  
  
    public static void main (String[] args) {  
  
        Main main = new Main();  
  
        Car car = new Car();  
        car.name = "Beetle";  
        car.model = "1300";  
        car.year = 1975;  
  
        main.setCar(car);  
  
        System.out.println(main.getCar().name);  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime null
- c) O código compila e imprime " "
- d) O código compila e imprime Beetle

**38) Quantos objetos estarão elegíveis para a coleta de lixo após a execução do construtor da classe Main:**

```
public class Main {  
  
    Main() {  
        int value = 0;  
        String name = "OCJP";  
        String version;  
        String number;  
        version = name;  
        number = version;  
        name = null;  
    }  
  
    public static void main (String[] args) {  
  
        new Main();  
  
        //<-----Objetos elegíveis  
    }  
}
```

- a) 0
- b) 1
- c) 2
- d) 3
- e) 4
- f) 5

**39) O que acontecerá ao tentarmos executar o programa a seguir?**

```
public class Main {  
  
    int positiveCount(int size, int...elements) {  
        int count = 0;  
        for (int number: elements) {  
            if (number >=0) {  
                count++;  
            }  
        }  
        return count;  
    }  
  
    public static void main(String...args) {  
        Main main = new Main();  
  
        int result = main.positiveCount(11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, -1, -2, -3);  
  
        System.out.println(result);  
    }  
}
```

- a) O programa não compila
- b) O programa compila e ao executar dispara uma exceção
- c) O programa compila e roda mas não imprime nada
- d) O programa compila, roda e imprime o valor 11
- e) O programa compila, roda e imprime o valor 10

**40) O que acontecerá ao tentarmos executar o programa a seguir?**

```
public class Main {  
  
    static boolean parImpar(String result) {  
        System.out.println(result);  
        return true;  
    }  
  
    public static void main(String...args) {  
  
        int num1 =4;  
        int num2 = 7;  
  
        if (num2 % 2 == 0 && parImpar("Impar")) {  
            return;  
        }  
        else if (num1 % 2 == 0 && parImpar("Par")) {  
            return;  
        }  
    }  
}
```

- a) O programa não compila
- b) O programa compila e imprime par
- c) O programa compila e imprime par Impar
- d) O programa compila e imprime Impar Par
- e) O programa compila e imprime Impar

**41) O que acontecerá ao tentarmos executar o programa a seguir?**

```
public class Main {  
  
    public static void main(String...args) {  
  
        System.out.println("SUM = " + 100 + 200 * 2);  
    }  
}
```

- a) O programa não compila
- b) O programa compila e imprime SUM = 100400
- c) O programa compila e imprime SUM = 500
- d) O programa compila e imprime SUM = 600
- e) O programa compila e imprime SUM = 1002002

**42) Ao executar o código a seguir, quantos objetos do tipo String serão criados?**

```
public class Main {  
  
    public static void main(String...args) {  
  
        String name = "Java";  
        String lowName = "java";  
        String upperName = new String("Java");  
        String javaName = "java";  
  
    }  
}
```

- a) 1
- b) 2
- c) 3
- d) 4

**43) O que acontecerá ao tentarmos executar o programa a seguir?**

```
public class Main {  
  
    public static void main(String...args) {  
  
        menu(100);  
  
    }  
  
    static public void menu(int choice) {  
  
        int OPTION_1 = 100;  
        int OPTION_2 = 200;  
        int OPTION_3 = 300;  
  
        switch(choice) {  
  
            case OPTION_1:  
                System.out.println("Option1");  
            case OPTION_2:  
                System.out.println("Option1");  
            case OPTION_3:  
                System.out.println("Option1");  
            default:  
                System.out.println("Not selected");  
  
        }  
  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime Option1
- c) O código compila e imprime Not Selected
- d) O código compila e imprime Option1 Option2 Option3 Not Selected

**44) O que acontecerá ao tentarmos executar o programa a seguir?**



```
class Person {
    String name = "empty";
    int maxAge = 0;

    Person() {
        name = "Default";
        maxAge = 130;
    }
}

public class Main {

    public static void main(String...args) {

        Person[] array = new Person[50];

        System.out.println(array[20].name.length());
    }
}
```

- a) O código não compila
- b) O código compila e imprime 5
- c) O código compila e imprime 7
- d) Nenhuma das opções anteriores

45) O que acontecerá ao tentarmos executar o programa a seguir? (Assuma que todos os imports foram realizados corretamente)

```
public class Main {  
    public static void main(String...args) {  
        ArrayList<Integer> array = new ArrayList<Integer>();  
        final int SIZE = 5;  
        for (int count=0; count < SIZE; count++) {  
            array.add(count);  
        }  
        for (int count = array.size() -1; count >= 0; count--) {  
            System.out.println(array.remove(array.size() -1));  
        }  
    }  
}
```

- a) O código não compila
- b) O código compila, roda e dispara uma exceção do tipo `IndexOutOfBoundsException`
- c) O código compila e imprime 4 3 2 1 0
- d) O código compila e imprime 0 1 2 3 4

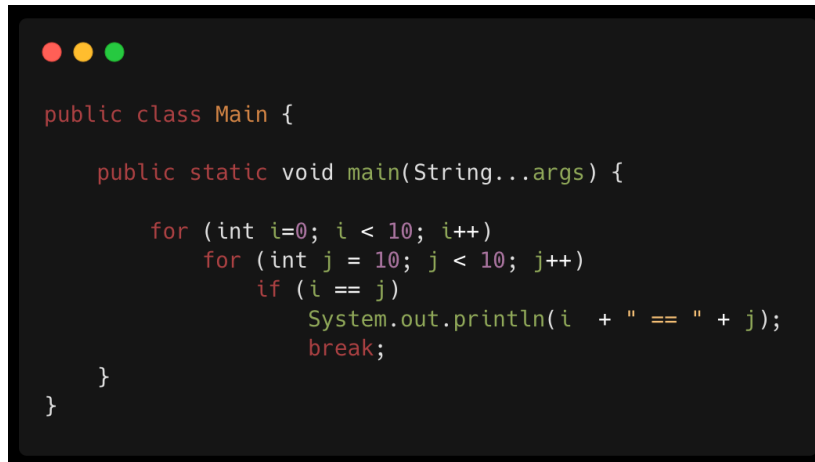
46) O que acontecerá ao tentarmos executar o programa a seguir?

```
class People {  
    String name;  
    String age;  
}  
  
public class Main {  
    public static void main(String...args) {  
        People[] array = new People[10];  
        for (People people: array) {  
            people = new People();  
        }  
        System.out.println(array[0].age);  
    }  
}
```



- a) O código não compila
- b) O código compila e dispara uma exceção do tipo NullPointerException
- c) O código compila e ao executar imprime o valor zero
- d) O código compila e ao executar imprime uma exceção do tipo IndexOutOfBoundsException

**47) O que acontecerá ao tentarmos executar o programa a seguir?**



```
public class Main {  
    public static void main(String...args) {  
        for (int i=0; i < 10; i++)  
            for (int j = 10; j < 10; j++)  
                if (i == j)  
                    System.out.println(i + " == " + j);  
                break;  
    }  
}
```

- a) O código não compila
- b) O código compila e ao executar não imprime nada
- c) O código compila e ao executar imprime 0 == 0 1 == 1 2 == 2 3 == 3 4 == 4 5 == 5 6 == 6 7 == 7 8 == 8 9 == 9
- d) O código compila e ao executar imprime apenas 0 == 0

48) O que acontecerá ao tentarmos executar o programa a seguir?

```
class Person {
    String name;
    String age;
    float height;

    void setHeight(float height) {
        this.height = height;
    }

    void setHeight(int height) {
        this.height = height;
    }

    void setHeight(double height) {
        this.height = (float)height;
    }
}

public class Main {

    public static void main(String...args) {

        Person person = new Person();

        person.setHeight(1.75);

        System.out.println(person.height);
    }
}
```

- a) O código não compila
- b) O código compila e imprime 1.75
- c) O código compila e imprime 0.0
- d) O código compila e ao executar dispara uma exceção

49) O que acontecerá ao tentarmos executar o programa a seguir?

```
package core;

public abstract class Car {
    protected float speed;

    abstract void setSpeed(float speed);

    abstract float getSpeed();
}

package certification;
import core.*;

class Ferrari extends Car {

    @Override
    void setSpeed(float speed) {
        this.speed = speed;
    }

    @Override
    float getSpeed() {
        return speed;
    }
}

public class Main {

    public static void main(String...args) {

        Car car = new Ferrari();

        car.setSpeed(300);

        System.out.println(car.getSpeed());
    }
}
```

- a) O código não compila
- b) O código compila e imprime 0.0
- c) O código compila e imprime 300.0
- d) O código compila e ao executar dispara uma exceção

**50) O que acontecerá ao tentarmos executar o programa a seguir?**

```
public class Main {  
    public static void main(String...args) {  
  
        ArrayList<Object> list = new ArrayList<Object>();  
  
        for (int i = 0; i < 10; i++) {  
            list.add(0, i);  
        }  
  
        for (Object value: list) {  
            System.out.println(value);  
        }  
    }  
}
```

- a) O código não compila
- b) O código compila e dispara uma exceção em tempo de execução
- c) O código compila e ao executar imprime 0 1 2 3 4 5 6 7 8 9
- d) O código compila e ao executar imprime 9 8 7 6 5 4 3 2 1 0

**51) O que acontecerá ao tentarmos executar o programa a seguir?**

```
public class Main {  
    public static void main(String...args) {  
  
        int[] array = null;  
  
        initArray(array);  
  
        System.out.println(array[4]);  
    }  
  
    public static void initArray(int[] array) {  
        array = new int[10];  
        array[4] = 100;  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime null
- c) O código compila e imprime 100
- d) O código compila e ao executar dispara uma exceção

52) Considere um método recursivo vazio, que não possui uma condição de parada, ao ser invocado qual o comportamento esperado para este programa?

```
public class Main {  
    public static void main(String...args) {  
        recursive();  
    }  
  
    public static void recursive() {  
        recursive();  
    }  
}
```

- a) O código não compila
- b) Compila, roda, não imprime nada e finaliza com sucesso
- c) Compila e ao rodar dispara uma exceção do tipo StackOverflowError
- d) Compila e ao rodar dispara uma exceção do tipo OutOfMemoryError
- e) Compila e ao rodar dispara uma exceção do tipo IndexOutOfBoundsException

53) O que acontecerá ao tentarmos executar o programa a seguir?

```
public class Main {  
    public static void main (String args[]) {  
        LocalDate date = LocalDate.of(2015, 2, 1);  
        date.withDayOfMonth(30);  
        System.out.println(date);  
    }  
}
```

- a) O código não compila
- b) O código compila e ao executar dispara uma exceção
- c) O código compila e ao executar imprime 2015-02-01
- d) O código compila e ao executar imprime 02-01-2015

**54) O Qual das palavras-chave a seguir não faz parte do conjunto de palavras reservadas da linguagem Java?**

- a) in
- b) goto
- c) strictfp
- d) instanceof
- e) transient

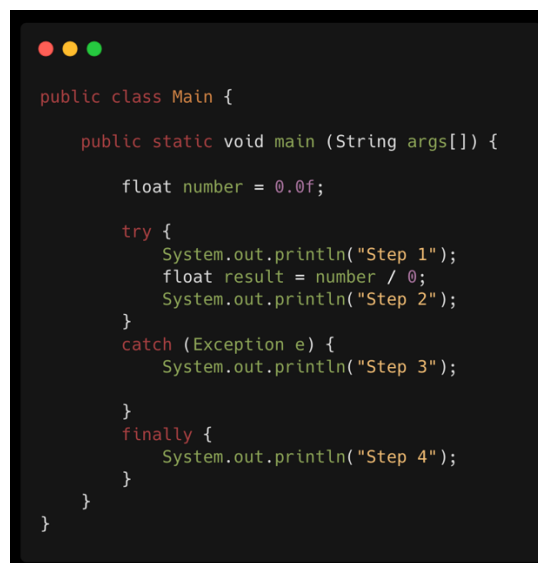
**55) Analise o código a seguir e assinale a opção correta após sua tentativa de execução?**



```
public class Main {  
    public static void main (String args[]) {  
        int count = 0;  
        Thread tread = new Thread(()->{  
            for ( ; count < 10; count++) {  
                System.out.println(count);  
            }  
        });  
    }  
}
```

- a) O código não compila
- b) O código compila e dispara uma exceção em tempo de execução
- c) O código compila, roda normalmente e não exibe nada
- d) O código compila, roda e exibe 0 1 2 3 4 5 6 7 8 9

**56) Analise o código a seguir e assinale a opção correta após sua tentativa de execução?**



```
public class Main {  
    public static void main (String args[]) {  
        float number = 0.0f;  
        try {  
            System.out.println("Step 1");  
            float result = number / 0;  
            System.out.println("Step 2");  
        }  
        catch (Exception e) {  
            System.out.println("Step 3");  
        }  
        finally {  
            System.out.println("Step 4");  
        }  
    }  
}
```

- a) O código não compila
- b) O código compila e imprime Step 1 Step 2 Step 3 Step 4
- c) O código compila e imprime Step 1 Step 3 Step 4
- d) O código compila e imprime Step 1 Step 2 Step 4