

Cuestións teóricas

1. O paso de parámetros, asociados a tipos de datos primitivos, ás funcións/métodos en Java son por:
 - a. **Valor**
 - b. Coincidencia
 - c. Referencia
 - d. Todas son falsas
2. Que son as variables locais?
 - a. **Aquelas variables que se declaran e só se poden empregar dentro dun método ou construtor. O seu tempo de vida é só a do método.**
 - b. Son aquelas variables que se definen dentro dunha clase e o seu período de vida é o asociado a unha clase. Poden empregarse dentro dos métodos da clase á que pertence.
 - c. Son aquelas variables que só se poden empregar dentro dun método a excepción dos construtores.
 - d. Todas son certas
3. Cal é o termo que máis se achega ao de Perm Gen en Java?
 - a. Heap space
 - b. Stack space
 - c. **Meta Space**
 - d. Pila
4. Cal é o cometido básico de empregar funcións/métodos/procedementos?
 - a. Simplificar a elaboración de programas complexos
 - b. Reutilizar código
 - c. Aplicar en implementación a técnica “*divide y vencerás*”
 - d. **Todas son certas**
5. Que paquete é importado por defecto nos ficheiros de código Java?
 - a. **java.lang**
 - b. java.base

- c. java.io
 - d. java.system
6. Indica o enunciado FALSO acerca dos paquetes de Java
- a. Non se aconsella crear clases asociadas ao paquete por defecto ou global, salvo para programiñas pequenos e de mostra.
 - b. proporcionan unha forma de nomear unha colección de clases (namespace)
 - c. axudan a que poidamos ter dúas clases co mesmo nome dentro do mesmo paquete
 - d. Cada paquete correspóndese cun directorio a nivel de sistema de arquivos
7. As clases creadas no default package ...
- a. Están directamente no raíz do proxecto
 - b. Non son visibles para o resto dos paquetes, clases, ...
 - c. Son visibles para o resto de clases do default-packages
 - d. Todas son certas
8. Cales son os niveis de visibilidade asociados a clases?
- a. Public e default
 - b. Public e package-private
 - c. Default e package-private
 - d. A e b son certas
9. Cantas clases (Top Level Class) podemos declarar nun arquivo *.java*?
- a. 1
 - b. Varias
 - c. 2
 - d. 3
10. Cantas clases públicas podemos declarar nun arquivo *.java*?
- a. 1
 - b. Varias
 - c. 2
 - d. 3

11. É obrigado declarar polo menos unha clase pública nun arquivo .java?
- a. Si e debe chamarse igual que o arquivo .java
 - b. Non
 - c. Non, pero unha das clases debe chamarse igual que o arquivo .java
 - d. Si, pero só pode haber unha única clase definida
12. Cal das seguintes é unha Helper/Utility Class?
- a. Integer
 - b. Scanner
 - c. Math
 - d. String
13. Que obxectos, en tempo de execución se almacenan na stack (pila), e cales no heap (montón)?
- a. No heap almacénanse as clases do JRE e os obxectos. Na stack almacénanse os valores das variables primitivas e as referencias a obxectos do heap.
 - b. No heap almacénanse as referencias a obxectos e valores das variables primitivas. Na stack almacénanse as chamadas a métodos e os obxectos instanciados.
 - c. Na stack e no heap non se almacena información do programa en execución. Almacénase no MemSpace os obxectos, referencias e variables locais. Tamén se almacenan os elementos estáticos e a información inmutable.
 - d. Todas son falsas
14. Que é o PermGen?
- a. É un espazo especial do heap (montón) en onde a JVM garda a traza de metadatos das clases que foron cargadas.
 - b. A partir de Java 8 foi substituído polo MetaSpace
 - c. Ten definido un tamaño máximo por defecto que é limitado (sobre 64-82 MB)
 - d. Todas son certas
15. Que erro obtemos cando sobrepasamos o PermGen space?
- a. StackOverflowError
 - b. ExceptionError
 - c. OutOfMemoryError

- d. IOException
16. Cal dos seguintes é o acrónimo de FQCN?
- a. Field and Query Core Notations
 - b. Full-Time Query Class Notation
 - c. Fully Qualified Core Network
 - d. Fully Qualified Class Name
17. Cal é o FQCN da clase String?
- a. java.lang.String
 - b. String
 - c. java.base.String
 - d. Todas son certas
18. Unha variable static, en que zona de memoria se almacena?
- a. Heap
 - b. Stack
 - c. PermGen
 - d. Todas son certas
19. Cal dos seguintes NON é un tag de javadoc?
- a. @author
 - b. @version
 - c. @param
 - d. @tech
20. Cal sería o tag axeitado para indicar que un método está obsoleto?
- a. @see
 - b. @exception
 - c. @deprecated
 - d. @serial
21. Que é un array?
- a. Un conxunto desordenado de ítems do mesmo tipo

- b. Unha secuencia de ítems de diferentes tipos
- c. Unha secuencia de ítems do mesmo tipo
- d. Un conxunto desordenado de ítems de diferentes tipos

22. Un vector ...

- a. É un array dunha dimensión
- b. É unha matriz de dúas dimensións
- c. É un arreglo multidimensional
- d. Todas son falsas

23. Indica a saída do programa

```
public static void main(String[] args)

    System.out.println(func(5));
}

static void func(int i) {

    final int MAX_TIME;

    MAX_TIME = 60;

    return true;
}
```

- a. Nada
- b. 5
- c. Erro de compilación
- d. Erro de execución

24. Indica a saída do programa

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

    System.out.println(func(5));
}

static int func(int v) throws InterruptedException {

    final int MAX_TIME = 60;

    return v;
    Thread.sleep(MAX_TIME);
}
```

- a. Nada
- b. 5
- c. Erro de compilación

d. Erro de execución

25. Indica a saída do programa

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

    System.out.println(func(5));
}

static int func(int v) throws InterruptedException {

    final int MAX_TIME = 60;

    MAX_TIME = v;
    Thread.sleep(MAX_TIME);
    return v;
}
```

a. Nada

b. 5

c. Erro de compilación

d. Erro de execución

26. Indica a saída do programa

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

    func(144);
}

static void func(int v) {

    int target = v, sqrt = 1;
    while (++sqrt * sqrt != target);
    System.out.println("sqrt(" + target + ") = " + sqrt);
}
```

a. 12

b. 1

c. Erro de compilación

d. Erro de execución

27. Indica a saída do programa

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

    String s = "";
    int i = s.length();
    func(s, 1);
    System.out.println(s.substring(s.length()));
}

static void func(String a, int v) {
    a = "a";
    v = 1;
}
```

a. Nada

b. a

- c. Erro de compilación
- d. Erro de execución

28. Indica a saída do programa

```
public static void main(String[] args){  
  
    String s = "32815";  
    System.out.println(func(s).length);  
}  
  
static String[] func(String a) {  
  
    return (a.split(""));  
}
```

- a. 1
- b. 5
- c. Erro de compilación
- d. Erro de execución

29. Indica a saída do programa

```
public static void main(String[] args){  
  
    String s = "3 2    8 1 5";  
    System.out.println(func(s).length);  
}  
  
static String[] func(String a) {  
  
    return (a.split("\\s+"));  
}
```

- a. 3
- b. 5
- c. Erro de compilación
- d. Erro de execución

30. Indica a saída do programa

```
public static void main(String[] args){  
  
    String s = "julio;mosquera gonzález;acebedo;616199xxx;;julio.mosquera@edu.xunta.es";  
    System.out.println(func(s)[4]);  
}  
  
static String[] func(String a) {  
  
    return (a.split(";"));  
}
```

- a. Nada
- b. julio.mosquera@edu.xunta.es

c. Erro de compilación

d. Erro de execución

31. Indica a saída do programa

```
public static void main(String[] args){  
  
    int[] vector = {2, 3, 4};  
    func(vector);  
    System.out.println(Arrays.toString(vector));  
}  
  
static void func(int[] v) {  
  
    vector[0] = 0;  
    vector[1] = 2;  
    vector[2] = 3;  
}
```

a. [2, 3, 4]

b. [0, 2, 3]

c. Erro de compilación

d. Erro de execución

32. Indica a saída do programa

```
public static void main(String[] args){  
  
    System.out.println(func(new int[]{2, 3, 4})[2]);  
}  
  
static int[] func(int[] v) {  
  
    v[0] = 0;  
    v[1] = 2;  
    v[2] = 3;  
    return v;  
}
```

a. 4

b. 3

c. Erro de compilación

d. Erro de execución

33. Indica a saída do programa


```
public static void main(String[] args){  
  
    int vector[] = {1, 2, 3};  
  
    System.out.println(func(vector));  
}  
  
static int func(int[] v) {  
  
    v[0] = 4;  
    v[2] = v[0] + v[1];  
}
```

- a. 4
- b. 3
- c. Erro de compilación
- d. Erro de execución

34. Indica a saída do programa

```
public static void main(String[] args){  
  
    System.out.println(func());  
}  
  
static int func(int ... v) {  
  
    int suma = 0;  
    for (int a: v) {  
        suma += a;  
    }  
    return suma;  
}
```

- a. 0
- b. 1
- c. Erro de compilación
- d. Erro de execución

35. Indica a saída do programa

```
public static void main(String[] args){  
  
    int[][] matriz = {{1,2,3},{4,5,6},{7,8,9}};  
    func(matriz);  
}  
  
static void func(int[][] m) {  
  
    int F;  
  
    for(int i = 0; i < m.length; i++) {  
        F = 0;  
        for (int j = 0; j < m[i].length; j++) {  
            F += m[i][j];  
            System.out.printf(" %d", m[i][j]);  
        }  
        System.out.print(" " + F + " " + F / m.length);  
        System.out.println("");  
    }  
}
```

1 2 3 6 2
4 5 6 15 5
a. 7 8 9 24 8

1 2 3 6 1
4 5 6 15 4
b. 7 8 9 24 7
c. Erro de compilación
d. Erro de execución

36. Indica a saída do programa

```
public static void main(String[] args){  
  
    func(vector);  
    System.out.println(Arrays.toString(vector));  
}  
  
static void func(int[] v) {  
  
    v = new int[10];  
    Arrays.fill(v, 5);  
  
}
```

a. [5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
b. []
c. Erro de compilación
d. Erro de execución

37. Indica a saída do programa

```
public static void main(String[] args){  
  
    int[] vector = new int[0];  
  
    func(vector);  
    System.out.println(Arrays.toString(vector));  
}  
  
static void func(int[] v) {  
  
    v = new int[10];  
    Arrays.fill(v, 5);  
    System.out.println(Arrays.toString(v));  
  
}
```

a. [5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
[]
b. [5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
c. Erro de compilación

d. Erro de execución

38. Indica a saída do programa

```
public static void main(String[] args){  
  
    int[] vector = new int[10];  
  
    func(vector);  
    System.out.println(Arrays.toString(vector));  
}  
  
static void func(int[] v) {  
  
    Arrays.fill(v, 2);  
  
}
```

a. [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

b. [2, 2, 2, 2, 2, 2, 2, 2, 2, 2]

c. Erro de compilación

d. Erro de execución

39. Indica a saída do programa

```
public static void main(String[] args){  
  
    int[] vector = new int[5];  
  
    func(vector);  
    System.out.println(Arrays.toString(vector));  
}  
  
static void func(int[] v) {  
  
    Arrays.setAll(v, (indiceVector) -> indiceVector * 2);  
  
}
```

a. [0, 2, 4, 6, 8]

b. [0, 0, 0, 0, 0]

c. Erro de compilación

d. Erro de execución

40. Cal das seguintes liñas compila?

```
int n1 = 9_45L;  
int n2 = 0B23;  
int n3 = 0XFF;  
int n4 = 0b101;
```

a. Todas

b. Ningunha

c. A 3ª e a 4ª

d. A 1ª e a 2ª

41. Indica a saída do programa

```
public static void main(String[] args){  
  
    func();  
}  
  
static void func() {  
  
    int a[] = new int[] {1, 2 ,3 ,4 ,5};  
  
    //Arrays.setAll(a, (indiceVector) -> a[indiceVector] * 3);  
  
    for (int i = 0; i <= a.length; i++) {  
        System.out.printf("Índice: %d vale: %d %n", i, a[i]);  
    }  
}
```

- Índice: 0 vale: 1
- Índice: 1 vale: 2
- Índice: 2 vale: 3
- Índice: 3 vale: 4
- a. Índice: 4 vale: 5
- Índice: 0 vale: 0
- Índice: 1 vale: 1
- Índice: 2 vale: 2
- Índice: 3 vale: 3
- b. Índice: 4 vale: 4
- c. Erro de compilación
- d. Erro de execución

42. Este código, ao executarse arroxa o resultado 3 0 3. Se o volvo executar, que sucederá?

```
Random r = new Random(10);  
  
System.out.print(r.nextInt(10) + " ");  
System.out.print(r.nextInt(10) + " ");  
System.out.print(r.nextInt(10) + " ");
```

- a. Volverá amosar 3 0 3
- b. Amosará 3 valores entre 0 e 9
- c. Amosará 3 valores entre 1 e 10
- d. Amosará 3 valores entre 0 e 10

43. Indica a saída do programa

```
public static void main(String[] args){  
  
    System.out.println(func());  
}  
  
static int func() {  
  
    Random r = new Random(10);  
    int i = 0;  
    while(i+1< r.nextInt());  
    return i;  
}
```

- a. 0
- b. Será un valor entre 0 e 9
- c. Erro de compilación
- d. Erro de execución

44. Indica a saída do programa

```
public static void main(String[] args){  
  
    System.out.println(func());  
}  
  
static int func() {  
  
    Random r = new Random();  
    int i = 0;  
    while((i+1)++ > r.nextInt());  
    return i;  
}
```

- a. 1
- b. Será un valor entre 1 e 10
- c. Erro de compilación
- d. Erro de execución

45. Cal é o resultado da execución do seguinte código? Supoñemos que este código está almacenado nun arquivo chamado **Codigos_17**.

```
public class Codigos_17 {  
    public static void main(String[] args){  
        A a = new A();  
        B b = new B();  
        C c = new C();  
        System.out.println("Finish!!!");  
    }  
}  
  
class A {  
}  
class B {  
}  
class C {  
}
```

- a. Finish!!!
- b. Erro de compilación porque non pode haber máis dunha clase por arquivo
- c. Erro de compilación

46. Erro de execución