

# **Java Coding Interview**

## **Java Programming Questions and Answers**



## 1. Qual è l'output del seguente programma?

```
byte[] a = {1,2,3}; byte[] b = (byte[]) a.clone(); print(a == b)
```

**Risposta: Falso**

## **2. Java supporta funzioni virtuali?**

**Risposta: Sì, tutte le funzioni non statiche sono virtuali**

**3. È possibile acquisire un lock su una classe?**

**Risposta: Sì, utilizzando un metodo statico sincronizzato**

#### 4. Quale pattern utilizzeresti per migliorare il seguente codice?

```
if (object instanceof A) doSomethingA; if (object instanceof B)  
doSomethingB;
```

**Risposta: Possiamo usare il pattern Visitor**

## **5. Come fornire servizi web sicuri?**

**Risposta: Possiamo usare il protocollo HTTPS**

## **6. A cosa serve WSDL?**

**Risposta: WSDL viene utilizzato per descrivere la struttura della richiesta del client e della risposta del server**



**7. Java usa il passaggio per valore o per riferimento?**

**Risposta: Java usa il riferimento per gli oggetti e il valore per i tipi primitivi**

## 8. Qual è l'output del seguente codice?

```
public static void main(String[] args) {  
    Employee e = new Employee();  
    e.setName("PPPP");  
    int i = 99;  
    test0(e, i);  
    System.out.println(e.getName());  
    System.out.println(i);  
}  
  
public static void test0(Employee e, int i) {  
    e.setName("AAAA");  
    i = 100;  
}
```

**Risposta: AAAA (passaggio per riferimento), 99 (passaggio per valore)**

## **9. Qual è la differenza tra Web Server e Application Server?**

**Risposta: Un Web Server può ospitare solo applicazioni web. Un Application Server può ospitare sia applicazioni web che non web e supporta protocolli diversi da HTTP, come RMI e RPC.**

**10. Scrivi un codice per stampare il seguente pattern, dove n è il numero di righe**

```
*  
* *  
* * *  
* * * *
```

**Risposta:**

```
for(int i = 1; i <= n; i++) {  
    for(int j = 0; j < i; j++) {  
        System.out.print("*");  
    }  
    System.out.print("\n");  
}
```

## 11. Scrivi il codice per stampare il pattern sopra usando la ricorsione in Java

Risposta:

```
public static void printStar(int n) {  
    if (n <= 1) {  
        System.out.print("*");  
    } else {  
        printStar(n - 1);  
        for (int i = 0; i < n; i++) {  
            System.out.print("*");  
        }  
    }  
    System.out.print("\n");  
}
```

**12. Cos'è una variabile `transient` ?**

**Risposta: Una variabile `transient` non può essere serializzata.**

**13. `Iterator` è una classe o un'interfaccia? Qual è il suo scopo?**

**Risposta: `Iterator` è un'interfaccia utilizzata per iterare sugli elementi di una collezione.**

## 14. Quali sono le somiglianze e le differenze tra una classe astratta e un'interfaccia?

**Risposta:**

- **Differenze:**

- Una classe può implementare più interfacce, ma può estendere solo una classe astratta.
- Le interfacce possono definire solo metodi pubblici e costanti, mentre una classe astratta può avere metodi protetti e statici.

- **Somiglianze:**

- Non è possibile istanziare né una classe astratta né un'interfaccia.



## 15. Descrivi i principi della programmazione orientata agli oggetti (OOP)

**Risposta:**

- **Incapsulamento:** Fornisce un modo per nascondere dati e metodi.
- **Ereditarietà:** Consente il riutilizzo delle classi padre nelle classi figlie.
- **Polimorfismo:** Permette di sovraccaricare e sovrascrivere i metodi.

## **16. Quali specificatori di accesso sono disponibili in Java?**

**Risposta:**

- Public
- Protected
- Private
- Default

## 17. Descrivi le classi wrapper in Java

**Risposta:** Le classi wrapper permettono di incapsulare i tipi primitivi in oggetti. Esempi:

- `Boolean` → `java.lang.Boolean`
- `Byte` → `java.lang.Byte`
- `Character` → `java.lang.Character`
- `Double` → `java.lang.Double`
- `Float` → `java.lang.Float`
- `Integer` → `java.lang.Integer`
- `Long` → `java.lang.Long`
- `Short` → `java.lang.Short`
- `Void` → `java.lang.Void`

## 18. Qual è l'output del seguente programma?

```
public class Test {  
    public static void main(String[] args) {  
        int x = 3;  
        int y = 1;  
        if (x = y)  
            System.out.println("Not equal");  
        else  
            System.out.println("Equal");  
    }  
}
```

**Risposta: C. Un errore nella riga `if (x = y)` causa un errore di compilazione.**

**19. Cos'è una variabile di classe?**

**Risposta: Le variabili statiche sono chiamate variabili di classe.**

## 20. Qual è la differenza tra `instanceof` e `getClass()` ?

### Risposta:

- `instanceof` è un operatore, non una funzione.
- `getClass()` è un metodo della classe `java.lang.Object`.
- `o.getClass().getName().equals("java.lang.Math")` verifica se la classe di `o` è `java.lang.Math`.
- `(o instanceof java.lang.Math)` verifica se `o` è un'istanza di `java.lang.Math`.

## 21. Cos'è `Externalizable` ?

Risposta: `Externalizable` estende l'interfaccia `Serializable` . Permette di inviare e ricevere dati in formato compresso usando `writeExternal()` e `readExternal()` .

## 22. Quali sono gli stati identificativi di un Thread?

**Risposta:**

- **R** - Thread in esecuzione o pronto per essere eseguito.
- **S** - Thread sospeso.
- **CW** - Thread in attesa su una variabile di condizione.
- **MW** - Thread in attesa su un blocco monitor.
- **MS** - Thread sospeso in attesa su un blocco monitor.



**23. Quali sono alcune alternative all'ereditarietà?**

**Risposta: La delegazione è un'alternativa all'ereditarietà.**

## **24. Perché Java non supporta la sovraccarica degli operatori?**

**Risposta: Java ha imparato da C++ che la sovraccarica degli operatori rende il codice più difficile da comprendere e mantenere.**

**25. `null` è una parola chiave in Java?**

**Risposta: No, il valore `null` non è una parola chiave.**

**26. Quali caratteri possono essere utilizzati come secondo carattere di un identificatore, ma non come primo?**

**Risposta: Le cifre da 0 a 9 non possono essere usate come primo carattere di un identificatore, ma possono essere usate dopo il primo carattere.**

## **27. Quanti bit sono usati per rappresentare Unicode, ASCII, UTF-16 e UTF-8?**

**Risposta:**

- Unicode richiede 16 bit.
- ASCII usa 7 bit (ma di solito è rappresentato con 8 bit).
- UTF-8 usa 8, 16 o 18 bit.
- UTF-16 usa pattern di 16 bit o più.

## 28. Quando scegliere una lista collegata rispetto a un array?

**Risposta:** Dipende da:

- **Memoria:** Array ha memoria preallocata, lista collegata usa memoria dinamica.
- **Accesso:** Array ha accesso  $O(1)$ , lista collegata  $O(n)$ .
- **Modifiche:** Array richiede shifting, lista collegata no.

**29. Qual è la differenza tra `Vector` e `ArrayList` ?**

**Risposta: `Vector` è sincronizzato, `ArrayList` no.**

### 30. Qual è la differenza tra coda ( Queue ) e pila ( Stack )?

Risposta:

- **Stack:** LIFO (Last In, First Out).
- **Queue:** FIFO (First In, First Out).

Ecco la traduzione:



## 31. Qual è la differenza tra array e ArrayList?

**Risposta:** Un `ArrayList` è ridimensionabile, un array no. Un `ArrayList` può memorizzare qualsiasi oggetto, mentre un array può contenere solo oggetti dello stesso tipo.

## 32. Puoi limitare la capacità iniziale di un `Vector` in Java?

**Risposta:** Sì, utilizzando il seguente costruttore:

```
public Vector(int capacity)
```

**33. Qual è la differenza tra `Enumeration` e `Iterator`?**

**Risposta:** `Iterator` ha un metodo `remove()`, mentre `Enumeration` no.

`Enumeration` fornisce solo un'interfaccia di lettura.

**34. Il codice nel blocco `finally` viene eseguito se si verifica un'eccezione e c'è un'istruzione `return` nel blocco `catch`?**

**Risposta:** Il blocco `finally` viene eseguito indipendentemente dal fatto che venga lanciata o catturata un'eccezione. Se un'eccezione si verifica e c'è un'istruzione `return` nel blocco `catch`, il blocco `finally` verrà comunque eseguito. L'unico caso in cui il `finally` non viene eseguito è quando il processo termina o esaurisce la memoria.

## 35. Crea una classe per la quale possono essere create al massimo due istanze

Risposta:

```
public class MyClass {  
    static int counter = 0;  
  
    MyClass() throws Exception {  
        counter++;  
        if (counter > 2) {  
            throw new Exception();  
        }  
    }  
}
```

### 36. Quali delle seguenti non sono parole chiave di Java?

- `if`
- `then`
- `end`
- `while`
- `case`

**Risposta:** `then` , `end`

## 37. Quali dei seguenti non sono identificatori Java validi?

- `4ofAU`
- `GiveMe6`
- `_whatQuiz`
- `_2030_`
- `$intheATM`
- `#dahuk`

**Risposta:** `4ofAU` , `#dahuk`

### 38. Quale delle seguenti righe compilerà senza avvisi o errori?

- `float f = 3.164;`
- `char c = "c";`
- `boolean b = null;`
- `int i = 10;`
- `int j = 10.0;`

**Risposta:** `int i = 10;`



**39. Quale delle seguenti istruzioni stamperà 19.0 ?**

- `System.out.println(Math.floor(19.7));`
- `System.out.println(Math.round(19.7));`
- `System.out.println(Math.ceil(19.7));`
- `System.out.println(Math.abs(19.7));`

**Risposta:** `System.out.println(Math.floor(19.7));`

## 40. Cosa succede se provi a compilare ed eseguire il seguente codice?

```
public class Test {  
    public static void main(String args[]) {  
        int[] myArray = new int[5];  
        System.out.println(myArray[0]);  
    }  
}
```

- A. Errore: `myArray` viene referenziato prima di essere inizializzato
- B. `null`
- C. `0`
- D. `5`

Risposta: `C`

## 41. Quali delle seguenti affermazioni sono vere?

- A. **ArrayList** è thread-safe.
- B. **Vector** è thread-safe.
- C. **ArrayList** e **Vector** sono entrambi **List**.
- D. Sia **Vector** che **ArrayList** utilizzano un array come struttura dati.
- E. **Vector** raddoppia la sua dimensione, mentre **ArrayList** aumenta la capacità solo della metà della dimensione attuale.
- F. **Vector** e **ArrayList** implementano entrambi **RandomAccess**.

**Risposta: B, C, D, E, F**

## 42. Qual è l'output del seguente programma?

```
import java.util.HashMap;  
import java.util.Map;  
public class MapTest {  
    public static void main(String aga[]){  
        Map m = new HashMap();  
        m.put(null, "Test");  
        m.put(null, "Fest");  
        System.out.println(m);  
    }  
}
```

- A. {null=Fest}
- B. **NullPointerException** alla riga 6
- C. {null=Test}
- D. Errore di compilazione alle righe 7 e 8
- E. Errore di compilazione alla riga 8, perché non si può sovrascrivere il valore "Test"

### 43. Quali delle seguenti affermazioni sono vere?

- A. **Hashtable** è sincronizzata, mentre **HashMap** non lo è.
- B. **Hashtable** non consente chiavi o valori nulli, mentre **HashMap** permette una chiave nulla e più valori nulli.
- C. **Hashtable** è una classe legacy.
- D. L'**Iterator** in **HashMap** è **fast-fail**, mentre l'**Enumerator** in **Hashtable** è **fail-safe**.

**Risposta: Tutte sono vere**

**44. Quali delle seguenti collezioni sono associate con Enumeration?**

- A. Vector
- B. HashMap
- C. Hashtable
- D. SortedSet

**Risposta: A e C**

## 45. Dato il seguente array 2D

```
int[][] myarray = new int[2][3];
```

Quale codice inizializza correttamente tutti gli elementi della seconda riga a 11?

A. `myarray[1] = {11,11,11};`

B. `myarray[1][0] = myarray[1][1] = myarray[1][2] = 11;`

C.

```
myarray[1][0] = 11;  
myarray[1][1] = 11;  
myarray[1][2] = 11;
```

D. `myarray[1] = new {11,11,11};`

E.

```
for (int i=0; i<myarray[1].length; i++)
```

**46. Una classe deve sempre definire almeno un costruttore?**

**A. Vero**

**B. Falso**

**Risposta: B**



## 47. Quale sarà l'output del seguente programma?

```
class A {  
    final public int GetResult(int a, int b) { return 0; }  
}  
class B extends A {  
    public int GetResult(int a, int b) { return 1; }  
}  
public class Test {  
    public static void main(String args[]) {  
        B b = new B();  
        System.out.println("x = " + b.GetResult(0, 1));  
    }  
}
```

- A. x = 0
- B. x = 1
- C. Errore di compilazione.
- D. Eccezione a runtime.

## 48. Quale sarà l'output del programma?

```
class Test {  
    public static void main(String[] args) {  
        Test s = new Test();  
        s.start();  
    }  
  
    void start() {  
        int a = 4;  
        int b = 5;  
        System.out.print(" " + 8 + 3 + " ");  
        System.out.print(a + b);  
        System.out.print(" " + a + b + " ");  
        System.out.print(foo() + a + b + " ");  
        System.out.println(a + b + foo());  
    }  
  
    String foo() {  
        return "foo";  
    }  
}
```

## 49. Quale sarà l'output del programma?

```
class myArray {
    boolean[] b = new boolean[3];
    int count = 0;

    void set(boolean[] x, int i) {
        x[i] = true;
        ++count;
    }

    public static void main(String[] args) {
        myArray ma = new myArray();
        ma.set(ma.b, 0);
        ma.set(ma.b, 2);
        ma.test();
    }

    void test() {
        if (b[0] && b[1] | b[2]) count++;
        if (b[1] && b[(++count - 2)]) count += 7;
        System.out.println("count = " + count);
    }
}
```

## 50. Quale sarà l'output del programma?

```
public void foo(boolean a, boolean b) {  
    if (a) {  
        System.out.println("A");  
    } else if (a && b) {  
        System.out.println("A && B");  
    } else {  
        if (!b) {  
            System.out.println("not B");  
        } else {  
            System.out.println("ELSE");  
        }  
    }  
}
```

- A. Se a è true e b è true , output: "A && B"
- B. Se a è true e b è false , output: "not B"
- C. Se a è false e b è true , output: "ELSE"
- D. Se a è false e b è false , output: "ELSE"

Ecco la traduzione delle ultime 10 domande e risposte:

## 51. Quale sarà l'output?

```
Float f = new Float("12");  
switch (f) {  
    case 12: System.out.println("Dodici");  
    case 0: System.out.println("Zero");  
    default: System.out.println("Predefinito");  
}
```

- A. Zero
- B. Dodici
- C. Predefinito
- D. Errore di compilazione

**Risposta: D (Solo numeri interi possono essere usati nello switch).**

## 52. Quale sarà l'output del programma?

```
class MyThread extends Thread {  
    public static void main(String[] args) {  
        MyThread t = new MyThread();  
        t.start();  
        System.out.print("uno. ");  
        t.start();  
        System.out.print("due. ");  
    }  
    public void run() {  
        System.out.print("Thread ");  
    }  
}
```

- A. Errore di compilazione
- B. Un'eccezione viene lanciata a runtime
- C. Stampa "Thread uno. Thread due."
- D. L'output non può essere determinato

## 53. Quale sarà l'output del programma?

```
class Test implements Runnable {  
    int x, y;  
    public void run() {  
        for (int i = 0; i < 1000; i++)  
            synchronized(this) {  
                x = 16;  
                y = 16;  
            }  
        System.out.print(x + " " + y + " ");  
    }  
    public static void main(String args[]) {  
        Test run = new Test();  
        Thread t1 = new Thread(run);  
        Thread t2 = new Thread(run);  
        t1.start();  
        t2.start();  
    }  
}
```



## 54. Quale sarà l'output del programma?

```
public class Test {  
    public static void main(String[] args) {  
        String s = "52";  
        try {  
            s = s.concat(".4");  
            double d = Double.parseDouble(s);  
            s = Double.toString(d);  
            int x = (int) Math.ceil(Double.valueOf(s).doubleValue());  
            System.out.println(x);  
        } catch (NumberFormatException e) {  
            System.out.println("Numero non valido");  
        }  
    }  
}
```

- A. 52
- B. 52.5
- C. 53

## 55. Quale sarà l'output del programma?

```
public class Test {  
    public static void stringReplace(String text) {  
        text = text.replace('j', 'k');  
    }  
    public static void bufferReplace(StringBuffer text) {  
        text.append("k");  
    }  
    public static void main(String args[]) {  
        String textString = new String("java");  
        StringBuffer textBuffer = new StringBuffer("java");  
        stringReplace(textString);  
        bufferReplace(textBuffer);  
        System.out.println(textString + textBuffer);  
    }  
}
```

- A. "java"
- B. "javak"

## 56. Quale sarà l'output del programma?

```
public class Test {  
    public static void main(String[] args) {  
        Boolean b1 = new Boolean("false");  
        boolean b2;  
        b2 = b1.booleanValue();  
        if (!b2) {  
            b2 = true;  
            System.out.print("a");  
        }  
        if (b1 & b2) {  
            System.out.print("b ");  
        }  
        System.out.println("c");  
    }  
}
```

- A. "c"
- B. "a c"

## 57. Quale riga è un esempio di uso inappropriato delle asserzioni?

```
public class Test {  
    public static int x;  
    public static int foo(int y) {  
        return y * 2;  
    }  
    public static void main(String[] args) {  
        int z = 6;  
        assert z > 0; /* Linea 11 */  
        assert z > 2 : foo(z); /* Linea 12 */  
        if (z < 7)  
            assert z > 4; /* Linea 14 */  
  
        switch (z) {  
            case 4: System.out.println("4 "); case 6: System.out.println("6 ");  
            default: assert z < 10;  
        }  
  
        if (z < 10)  
            assert z > 4 : z++; /* Linea 22 */  
        System.out.println(z);  
    }  
}
```

## 58. Quale sarà l'output del programma?

```
public class Test {  
    public static int y;  
    public static void foo(int x) {  
        System.out.print("foo ");  
        y = x;  
    }  
    public static int bar(int z) {  
        System.out.print("bar ");  
        System.out.print("bar ");  
        return y = z;  
    }  
    public static void main(String[] args) {  
        int t = 0;  
        assert t > 0 : bar(7);  
        assert t > 1 : foo(8); /* Linea 18 */  
        System.out.println("done ");  
    }  
}
```

**59. Quale affermazione è vera per `java.util.ArrayList` ?**

- A. Gli elementi nella collezione sono ordinati.
- B. La collezione è garantita come immutabile.
- C. Gli elementi nella collezione sono garantiti come unici.
- D. Gli elementi nella collezione sono accessibili tramite una chiave univoca.

**Risposta: A**

## 60. Quale linea di codice è adatta per avviare un thread?

```
class X implements Runnable {  
    public static void main(String args[]) {  
        /* Completare la funzione */  
    }  
    public void run() {}  
}
```

- A. `Thread t = new Thread(X);`
- B. `Thread t = new Thread(X); t.start();`
- C. `X run = new X(); Thread t = new Thread(run); t.start();`
- D. `Thread t = new Thread(); x.run();`

**Risposta: C**