
Qual e' il modo corretto per creare una classe `MiaClasse` di tipo Generics, parametrizzata con un tipo in input? [1]

1. `class MiaClasse<T>`
2. `class MiaClasse<Generics>`
3. `class MiaClasse(Type Generics)`
4. `Generics MiaClasse`

Quale e' l'espressione regolare corretta per validare una stringa contentente uno o piu caratteri qualsiasi tra "a" oppure "b" oppure "d"? [1]

1. [abd]+
2. [abd]*
3. [abd]{2,6}
4. [abd]{4}

Qual e' il modo corretto per ottenere un Enumeration da un Vector, creato con `Vector<String> vector = new Vector<String>()`? [1]

1. `Enumeration<String> en = vector.elements();`
2. `Enumeration<String> en = new Enumeration(vector);`
3. `Enumeration<String> en = Enumeration.createFrom(vector);`
4. `Enumeration<String> en = vector.elements(new Enumeration());`

Quando un metodo si dice ricorsivo? [1]

1. Quando al suo interno contiene una chiamata al metodo stesso
2. Quando al suo interno contiene un ciclo for o while
3. Quando al suo interno contiene almeno due costrutti for, while o if annidati
4. Quando viene richiamato dal metodo main

Quale e' lo speciale tag che permette di descrivere un valore di ritorno in javadoc? [1]

1. `@return`
2. `#return`
3. `@returns`
4. `@ret`

Quale e' l'espressione regolare corretta per validare una stringa contentente 6 caratteri qualsiasi tra "a" oppure "b" oppure "c"? [1]

1. [abc]{6}
2. [a-d]{6}
3. [abc][6]
4. [a-c]{0,6}

Qual e' il modo corretto per creare una istanza della class `MiaClasse` di tipo Generics parametrizzata con un tipo in input? [1]

1. `MiaClasse<Integer> miaIstanza = new MiaClasse<Integer>();`

2. `MiaClasse<Generics> miaIstanza = new MiaClasse<Integer>();`
3. `MiaClasse(Type Integer) miaIstanza = new MiaClasse(Type Integer);`
4. `MiaClasse miaIstanza = Generics MiaClasse();`

Quali tra i seguenti frammenti di codice sono corretti per dichiarare una nuova annotazione custom? [1]

1. `public @interface checkAnnotation {}`
2. `public @annotation checkAnnotation {}`
3. `public annotation checkAnnotation {}`
4. `public interface checkAnnotation {}`

Cosa permette di fare il codice riportato di seguito?

```
BufferedReader input = new BufferedReader(new FileReader("file.txt"));
String str;
while( (str = input.readLine()) != null)
  System.out.println(str);
```

[1]

1. Permette di aprire il file "file.txt" in lettura e di leggere l'intero file
2. Permette di aprire il file "file.txt" in lettura e di leggere la prima riga
3. Permette di aprire il file "file.txt" in scrittura e scrivere una riga vuota
4. Permette di aprire il file "file.txt" in lettura e di contare quanti caratteri "a capo" ci sono

---

Quale e' l'espressione regolare piu' corretta per validare una data nella forma mm/dd/aaaa? [3]

1. (0[1-9]|1[012])/(0[1-9]|[12][0-9]|3[01])/[0-9]{4}
2. (0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[012])/[0-9]{4}
3. [0-9]{4}/(0[1-9]|[12][0-9]|3[01])/(0[1-9]|1[012])
4. (0[1-9]|1[0123])/(0[1-9]|[12][0-9]|3[01])/[0-9]{4}

Quale e' il codice corretto per recuperare l'attributo di istanza "numZampe" dalla classe "Animale"? [2]

1. `Class<Animale> classAnimale = Animale.class; try{Field fieldNumZampe = classAnimale.getField("numZampe");}catch(NoSuchFieldException e){}`
2. `Class<Animale> classAnimale = Animale.class; Field fieldNumZampe = classAnimale.getField("numZampe");`
3. `Class<Animale> classAnimale = Animale.class; try{Field fieldNumZampe = classAnimale.getField("numZampe");}catch(NoSuchMethodException e){}`
4. `Class<Animale> classAnimale = Animale.class; try{Field fieldNumZampe = classAnimale.getMethod("numZampe");}catch(NoSuchMethodException e){}`

Quale e' la frase che descrive in maniera migliore il funzionamento dell'annotazione "@override"? [2]

1. @Override e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta sta effettuando un overwriting di un metodo della sua superclasse
2. @Override e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta sta effettuando un overloading di un metodo della sua superclasse
3. @Override e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta ridefinisce il corrispondente attributo ereditato dalla superclasse
4. @Override e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta sta effettuando un overloading di un attributo della sua superclasse

---

Cosa permette di fare il codice riportato di seguito?
```
BufferedReader input = new BufferedReader(new FileReader("file.txt"));
String str = input.readLine();
```
[1]

1. Permette di aprire il file "file.txt" in lettura e di leggere la prima riga
2. Permette di aprire il file "file.txt" in lettura e di leggere l'intero file
3. Permette di aprire il file "file.txt" in scrittura e scrivere una riga vuota
4. Permette di aprire il file "file.txt" in scrittura in modalita' append

---

Quale e' lo speciale tag che permette di descrivere un parametro in javadoc? [1]

1. `@param`
2. `#param`
3. `@params`
4. `@par`

Che valore restituisce la chiamata `mioMetodo("abcde")`, se il metodo e' implementato come segue?
```
public int mioMetodo(String str) {
  if(str.equals(""))
    return 0;
  else
    return 1 + mioMetodo(str.substring(1, str.length()));
}
```
[2]

1. 5
2. Il metodo entra in un loop di chiamate infinito
3. 0
4. 1

Quale e' il modo corretto per scrivere commenti documentali? [1]

1. ```
   /**
       * Commenti sulla variabile miaVariabile, un intero qualsiasi.
       */
   ```
2. ```
   /*
       * Commenti sulla variabile miaVariabile, un intero qualsiasi.
       */
   ```
3. ```
   /*
       Commenti sulla variabile miaVariabile, un intero qualsiasi.
       */
   ```
4. ```
   //
       * Commenti sulla variabile miaVariabile, un intero qualsiasi.
       //
   ```

Cosa permette di fare il codice riportato di seguito?
```
BufferedWriter output = new BufferedWriter(new FileWriter("file.txt", true));
```
[1]

1. Permette di aprire il file "file.txt" in scrittura in modalita' append
2. Permette di aprire il file "file.txt" in scrittura all'inizio del file
3. Permette di aprire il file "file.txt" in lettura
4. Permette di rimuovere il file

Cosa permette di fare il codice riportato di seguito?

`BufferedWriter output = new BufferedWriter(new FileWriter("file.txt", false));` [1]

1. Permette di aprire il file "file.txt" in scrittura all'inizio del file
2. Permette di aprire il file "file.txt" in scrittura in modalita' append
3. Permette di aprire il file "file.txt" in lettura
4. Permette di rimuovere il file

Quale e' il codice corretto per recuperare il metodo "aumentaVelocita" dalla classe "Bici" che prevede un parametro in input di tipo "int"? [2]

1. ```
   Class<Bici> classBici = Bici.class; try{Method methodAumentaVelocita =
   classBici.getMethod("aumentaVelocita", new Class[]
   {int.class});}catch(NoSuchMethodException e){}
   ```
2. ```
   Class<Bici> classBici = Bici.class; try{Method methodAumentaVelocita =
   classBici.getMethod("aumentaVelocita", new Class[]
   {});}catch(NoSuchMethodException e){}
   ```
3. ```
   Class<Bici> classBici = Bici.class; Method methodAumentaVelocita =
   classBici.getMethod("aumentaVelocita", new Class[] {String.class});
   ```
4. ```
   Class<Bici> classBici = Bici.class; try{Method methodAumentaVelocita =
   classBici.getMethod("aumentaVelocita", new Class[] {String.class,
   int.class});}catch(NoSuchFieldException e){}
   ```

Quale e' l'espressione regolare corretta per validare una stringa contenente un qualsiasi carattere minuscolo dell'alfabeto? [1]

1. [a-z]
2. [abcdefgilmvz]
3. [0-9]
4. [a-zA-Z]

Cosa restuisce il metodo `next()` chiamato su una istanza della classe `Iterator`? [1]

1. il prossimo elemento dell'array da scorrere
2. un valore booleano che indica se ci sono altri elementi dell'array da scorrere
3. un valore booleano che indica se l'array e' vuoto
4. la lista completa su cui l'Iterator sta iterando

Quale e' la frase che descrive in maniera migliore il funzionamento dell'annotazione "@Deprecated"? [2]

1. @Deprecated e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta non dovrebbe piu' essere usato
2. @Deprecated e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta dovrebbe obbligatoriamente essere usato
3. @Deprecated e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta non puo' essere usato
4. @Deprecated e' una annotazione utilizzata per indicare al compilatore che il metodo su cui e' posta effettua l'overloading di un'altro metodo

---

Quale e' l'espressione regolare piu' corretta per validare un indirizzo email? [3]

1. [a-zA-z0-9._%-]+@[a-zA-z0-9._%-]+\.[a-zA-Z]{2,4}

2. [a-zA-z0-9._%-]+@[a-zA-z0-9._%-]+\.[a-zA-Z]{1,2}
3. [a-zA-z0-9._%-]*@[a-zA-z0-9._%-]*\.[a-zA-Z]{2,4}
4. [a-zA-z0-9._%-]+@[a-zA-z0-9._%-]+.[a-zA-Z]{2,4}

Che valore restituisce la chiamata `calcola(0)`, se il metodo e' implementato come segue?

```
public int calcola(int n) {
  if(n < 0)
    return -1;
  else
    return calcola(n + 1);
}
```
[2]

1. Il metodo entra in un loop di chiamate infinito
2. 0
3. 1
4. -1

Quale tra le seguenti lambda expressions NON e' corretta, considerando l'interfaccia Condizione riportata di seguito?

```
interface Condizione {
  boolean test(String s);
}
```
[2]

1. `(String s) -> { char c = s.charAt(0) c != 'c'}`
2. `(s) -> s != null && s.length() > 0`
3. `(String s) -> { char c = s.charAt(0); return c == 'a';}`
4. `(s) -> false`

Quale tra questi e' il modo corretto per creare una istanza della classe `LocalDate`? [1]

1. `LocalDate data = LocalDate.of(2005, 3, 6);`
2. `LocalDate data = LocalDate.new(2005, 3, 6);`
3. `LocalDate data = new LocalDate(2005, 3, 6);`
4. `LocalDate data = new LocalDate(2005, Month.MARCH, 6);`

Cosa restuisce il metodo `hasNext()` chiamato su una istanza della classe `Iterator`? [1]

1. un valore booleano che indica se ci sono altri elementi dell'array da scorrere
2. il prossimo elemento dell'array da scorrere
3. un valore booleano che indica se l'array e' vuoto
4. la lista completa su cui l'Iterator sta iterando

Quale e' il codice corretto per recuperare il costruttore dalla classe "Bici" che prevede un parametro in input di tipo "int"? [1]

1. `Class<Bici> classBici = Bici.class; try{Constructor method = classBici.getConstructor(new Class[] {int.class});}catch(NoSuchMethodException e){}`
2. `Class<Bici> classBici = Bici.class; try{Constructor costructor = classBici.getConstructor(new Class[] {String.class});}catch(NoSuchMethodException e){}`
3. `Class<Bici> classBici = Bici.class; try{Constructor costructor = classBici.getConstructor(new Class[] {});}catch(NoSuchMethodException e){}`

4. `Class<Bici> classBici = Bici.class; try{Constructor costructor = classBici.getConstructor("Bici", new Class[] {int.class});}catch(NoSuchMethodException e){}`

I commenti documentali a chi possono essere riferiti? [1]

1. classi, metodi e attributi
2. solo a classi e metodi
3. solo a classi e attributi
4. solo a metodi e attributi

Quale sara' il valore della variabile `total` dopo aver eseguito il codice sotto riportato?

```
List<String> arrStringhe = new ArrayList<String>();
arrStringhe.add("AAA");
arrStringhe.add("BBB");
arrStringhe.add("CCC");
String total = "ZZZ";
for(String s : arrStringhe)
  total += s;
```

[2]

1. ZZZAAABBBCCC
2. AAABBBCCC
3. ZZZ
4. Nessuna delle risposte precedenti