

Linguaggi di Programmazione I – Java-4

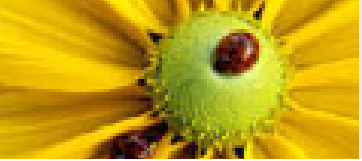
Prof. Marco Faella

<mailto://m.faella@unina.it>

<http://wpage.unina.it/mfaella>

Materiale didattico elaborato con i Proff. Sette e Bonatti

4 aprile 2023

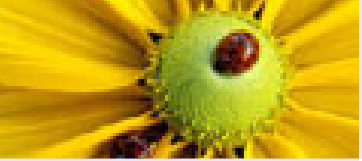


Complementi

String e StringBuffer

Garbage collection

Questionario



String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

String e StringBuffer



String (1)

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Le stringhe sono oggetti che possono essere creati come segue:

```
String s = new String("abcdef");
```



String (1)

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

- Le stringhe sono oggetti che possono essere creati come segue:

```
String s = new String("abcdef");
```

oppure così:

```
String s = "abcdef";
```

- Vedremo tra poco quali sono le (sottili) differenze tra questi modi.



String (1)

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

- Le stringhe sono oggetti che possono essere creati come segue:

```
String s = new String("abcdef");
```

oppure così:

```
String s = "abcdef";
```

- Vedremo tra poco quali sono le (sottili) differenze tra questi modi.



String (1)

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Le stringhe sono oggetti che possono essere creati come segue:

```
String s = new String("abcdef");
```

oppure così:

```
String s = "abcdef";
```

- Vedremo tra poco quali sono le (sottili) differenze tra questi modi.

- Nota: "abcdef" si chiama *letterale String*
- Analogamente, 1.0 si chiama *letterale double*



Immutabilità

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Le stringhe sono oggetti **immutabili**
- Immutabilità significa che, una volta assegnato all'oggetto un contenuto, esso è fissato per sempre (capiremo tra poco anche perché deve essere così).



Immutabilità

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

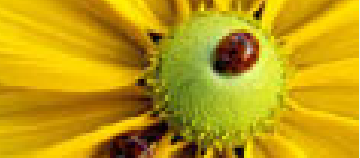
StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Le stringhe sono oggetti **immutabili**
- Immutabilità significa che, una volta assegnato all'oggetto un contenuto, esso è fissato per sempre (capiremo tra poco anche perché deve essere così).
- Attenzione: immutabili sono gli oggetti `String`, non i loro riferimenti. Questi ultimi possono cambiare valore.
- Anche i tipi wrapper sono immutabili



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

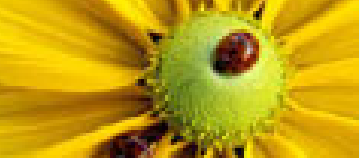
[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

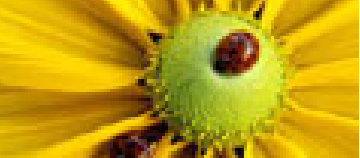
[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");  
System.out.println(s);    // cosa stampa?
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```

```
String x = "Walter";
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("□White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```

```
String x = "Walter";  
x.concat("□White");
```




Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("□White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```

```
String x = "Walter";  
x.concat("□White");  
System.out.println(x);    // cosa stampa?
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

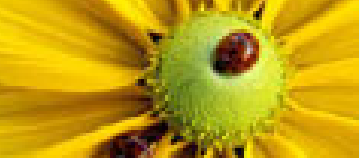
[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat(" White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```

```
String x = "Walter";  
x.concat(" White");  
System.out.println(x);    // cosa stampa?
```

```
String s1 = "A";  
String s2 = s1 + "B";  
s1.concat("C");  
s2.concat(s1);  
s1 += "D";                // Quanti oggetti in gioco?  
System.out.println(s1 + s2); // Cosa stampa?
```



Esempi

String e
StringBuffer

String (1)
Immutabilità

Esempi

String pool
StringBuffer (1)
Uguaglianza

Garbage collection

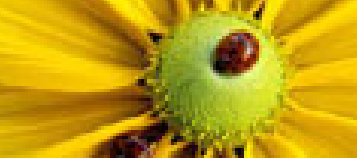
Questionario

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```

```
String x = "Walter";  
x.concat("_White");  
System.out.println(x);    // cosa stampa?
```

```
String s1 = "A";  
String s2 = s1 + "B";  
s1.concat("C");  
s2.concat(s1);  
s1 += "D";                // Quanti oggetti in gioco?  
System.out.println(s1 + s2); // Cosa stampa?
```

```
String s1 = "abc";  
String s2 = s1 + "";  
String s3 = "abc";  
System.out.println(s1 == s2);
```



Esempi

[String e
StringBuffer](#)

[String \(1\)](#)
[Immutabilità](#)

[Esempi](#)

[String pool](#)
[StringBuffer \(1\)](#)
[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);   // e qui?
```

```
String x = "Walter";  
x.concat("_White");  
System.out.println(x);    // cosa stampa?
```

```
String s1 = "A";  
String s2 = s1 + "B";  
s1.concat("C");  
s2.concat(s1);  
s1 += "D";                      // Quanti oggetti in gioco?  
System.out.println(s1 + s2);    // Cosa stampa?
```

```
String s1 = "abc";  
String s2 = s1 + "";  
String s3 = "abc";  
System.out.println(s1 == s2);   // false!  
System.out.println(s1 == s3);
```



Esempi

String e
StringBuffer

String (1)
Immutabilità

Esempi

String pool
StringBuffer (1)
Uguaglianza

Garbage collection

Questionario

```
String s = "Walter";  
String s2 = s;  
s = s.concat("_White");  
System.out.println(s);    // cosa stampa?  
System.out.println(s2);    // e qui?
```

```
String x = "Walter";  
x.concat("_White");  
System.out.println(x);    // cosa stampa?
```

```
String s1 = "A";  
String s2 = s1 + "B";  
s1.concat("C");  
s2.concat(s1);  
s1 += "D";                // Quanti oggetti in gioco?  
System.out.println(s1 + s2); // Cosa stampa?
```

```
String s1 = "abc";  
String s2 = s1 + "";  
String s3 = "abc";  
System.out.println(s1 == s2);    // false!  
System.out.println(s1 == s3);    // true!
```



String **pool**

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

- Per motivi di efficienza, poichè nelle applicazioni i letterali `String` occupano molta memoria, la JVM riserva un'area speciale di memoria ad essi: la *String constant pool*.



String **pool**

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

- Per motivi di efficienza, poichè nelle applicazioni i letterali `String` occupano molta memoria, la JVM riserva un'area speciale di memoria ad essi: la *String constant pool*.
- Quando il compilatore incontra un letterale `String`, esso controlla se è già presente nel pool.



String **pool**

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Per motivi di efficienza, poichè nelle applicazioni i letterali `String` occupano molta memoria, la JVM riserva un'area speciale di memoria ad essi: la *String constant pool*.
- Quando il compilatore incontra un letterale `String`, esso controlla se è già presente nel pool.
 - ◆ Se è presente, allora il letterale viene interpretato come un riferimento all'oggetto `String` esistente
 - ◆ Altrimenti, viene creato un nuovo oggetto `String` e aggiunto al pool.



String pool

[String e
StringBuffer](#)

[String \(1\)](#)

[Immutabilità](#)

[Esempi](#)

[String pool](#)

[StringBuffer \(1\)](#)

[Uguaglianza](#)

[Garbage collection](#)

[Questionario](#)

- Per motivi di efficienza, poichè nelle applicazioni i letterali `String` occupano molta memoria, la JVM riserva un'area speciale di memoria ad essi: la *String constant pool*.
- Quando il compilatore incontra un letterale `String`, esso controlla se è già presente nel pool.
 - ◆ Se è presente, allora il letterale viene interpretato come un riferimento all'oggetto `String` esistente
 - ◆ Altrimenti, viene creato un nuovo oggetto `String` e aggiunto al pool.
- Questo meccanismo di condivisione può funzionare perché gli oggetti `String` sono immutabili.



String **pool**

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Per motivi di efficienza, poichè nelle applicazioni i letterali `String` occupano molta memoria, la JVM riserva un'area speciale di memoria ad essi: la *String constant pool*.
- Quando il compilatore incontra un letterale `String`, esso controlla se è già presente nel pool.
 - ◆ Se è presente, allora il letterale viene interpretato come un riferimento all'oggetto `String` esistente
 - ◆ Altrimenti, viene creato un nuovo oggetto `String` e aggiunto al pool.
- Questo meccanismo di condivisione può funzionare perché gli oggetti `String` sono immutabili. Se lo stesso letterale compare in punti diversi del codice, l'eventuale modifica di un letterale modificherebbe anche l'altro



String pool

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Per motivi di efficienza, poichè nelle applicazioni i letterali String occupano molta memoria, la JVM riserva un'area speciale di memoria ad essi: la *String constant pool*.
- Quando il compilatore incontra un letterale String, esso controlla se è già presente nel pool.
 - ◆ Se è presente, allora il letterale viene interpretato come un riferimento all'oggetto String esistente
 - ◆ Altrimenti, viene creato un nuovo oggetto String e aggiunto al pool.
- Questo meccanismo di condivisione può funzionare perché gli oggetti String sono immutabili. Se lo stesso letterale compare in punti diversi del codice, l'eventuale modifica di un letterale modificherebbe anche l'altro
- Qual è quindi la differenza tra i due enunciati?

```
String s = "abcdef";
```

```
String s = new String("abcdef");
```



StringBuffer (1)

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Se si deve fare un uso intensivo di manipolazione di stringhe, allora è opportuno usare le classi `StringBuilder` e `StringBuffer`: esse sono simili a `String`, ma sono *mutabili*
- La differenza tra le due è che `StringBuffer` è *thread-safe*



StringBuffer (1)

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

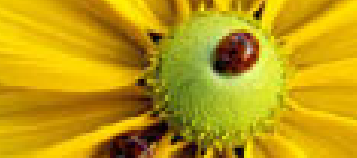
Garbage collection

Questionario

- Se si deve fare un uso intensivo di manipolazione di stringhe, allora è opportuno usare le classi `StringBuilder` e `StringBuffer`: esse sono simili a `String`, ma sono *mutabili*
- La differenza tra le due è che `StringBuffer` è *thread-safe*
- Per esempio:

```
StringBuffer s = new StringBuffer("Walter");  
s.append("_White");  
System.out.println(s); // cosa stampa?
```

```
String contenuto = s.toString();
```



StringBuffer (1)

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

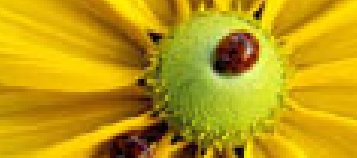
Questionario

- Se si deve fare un uso intensivo di manipolazione di stringhe, allora è opportuno usare le classi `StringBuilder` e `StringBuffer`: esse sono simili a `String`, ma sono *mutabili*
- La differenza tra le due è che `StringBuffer` è *thread-safe*
- Per esempio:

```
StringBuffer s = new StringBuffer("Walter");  
s.append("_White");  
System.out.println(s); // cosa stampa?  
  
String contenuto = s.toString();
```

- Attenzione:

```
StringBuffer s = "abc";  
// Illegale: String e StringBuffer  
// non sono auto-convertibili!
```



StringBuffer (1)

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- Se si deve fare un uso intensivo di manipolazione di stringhe, allora è opportuno usare le classi `StringBuilder` e `StringBuffer`: esse sono simili a `String`, ma sono *mutabili*
- La differenza tra le due è che `StringBuffer` è *thread-safe*
- Per esempio:

```
StringBuffer s = new StringBuffer("Walter");  
s.append("_White");  
System.out.println(s); // cosa stampa?  
  
String contenuto = s.toString();
```

- Attenzione:

```
StringBuffer s = "abc";  
    // Illegale: String e StringBuffer  
    // non sono auto-convertibili!  
  
StringBuffer s = (StringBuffer) "abc";  
    // Illegale: neanche con cast!
```



Uguaglianza

- String e StringBuffer
- String (1)
- Immutabilità
- Esempi
- String pool
- StringBuffer (1)
- Uguaglianza**
- Garbage collection
- Questionario

- **ATTENZIONE:** mentre la classe `String` sovrascrive il metodo `equals` in modo da controllare l'uguaglianza del contenuto dei due oggetti (quello corrente e quello ricevuto come parametro), le classi `StringBuffer` e `StringBuilder` non lo sovrascrivono ed usano quello ereditato da `Object`, che funziona come l'operatore `==` (cioè compara i riferimenti).



Uguaglianza

String e
StringBuffer

String (1)

Immutabilità

Esempi

String pool

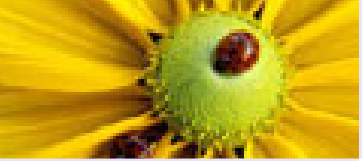
StringBuffer (1)

Uguaglianza

Garbage collection

Questionario

- **ATTENZIONE:** mentre la classe `String` sovrascrive il metodo `equals` in modo da controllare l'uguaglianza del contenuto dei due oggetti (quello corrente e quello ricevuto come parametro), le classi `StringBuffer` e `StringBuilder` non lo sovrascrivono ed usano quello ereditato da `Object`, che funziona come l'operatore `==` (cioè compara i riferimenti).
- Le classi `String`, `StringBuffer` e `StringBuilder` sono `final`. Esse non possono essere specializzate: sovrascriverne i metodi potrebbe creare problemi di sicurezza.



[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

[Esempio \(3\)](#)

[Esempio \(4\)](#)

[Esempio \(5\)](#)

[Questionario](#)

Garbage collection



Definizioni

[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

[Esempio \(3\)](#)

[Esempio \(4\)](#)

[Esempio \(5\)](#)

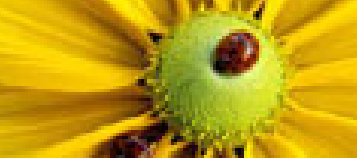
[Questionario](#)

- Rilascio automatico della memoria non più accessibile dal programma
- Un data object è “eleggibile” per la GC se non è più accessibile dal programma

Esempio:

```
public static void main(String[] args) {  
    StringBuffer sb = new StringBuffer("Ciao");  
    System.out.println(sb);  
    sb = null;  
    // ora l'oggetto StringBuffer e' eleggibile per la GC  
}
```

Nota: tranne casi eccezionali, le stringhe nello string pool non sono mai eleggibili per la GC



Algoritmo di GC

[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

[Esempio \(3\)](#)

[Esempio \(4\)](#)

[Esempio \(5\)](#)

[Questionario](#)

Algoritmo *mark-and-sweep*

1. Fase *mark*: a partire dallo stack e dalla regione statica, esplorare tutti gli oggetti accessibili e marcarli come tali
2. Fase *sweep*: rilasciare tutti i data object dello heap che non sono stati marcati come accessibili

La fase mark è analoga alla visita di un grafo: i nodi sono oggetti e gli archi sono riferimenti tra oggetti

Nota: la JVM contiene diversi algoritmi di GC alternativi



Esempio (2)

[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

[Esempio \(3\)](#)

[Esempio \(4\)](#)

[Esempio \(5\)](#)

[Questionario](#)

```
public static void main(String[] args) {  
    StringBuffer s1 = new StringBuffer("Ciao");  
    StringBuffer s2 = new StringBuffer("Addio");  
    System.out.println(s1);  
    // l'oggetto riferito da s1 non e' ancora  
    // eleggibile per GC
```



Esempio (2)

[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

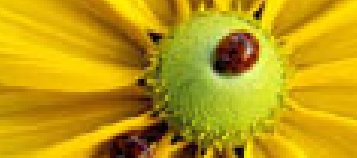
[Esempio \(3\)](#)

[Esempio \(4\)](#)

[Esempio \(5\)](#)

[Questionario](#)

```
public static void main(String[] args) {  
    StringBuffer s1 = new StringBuffer("Ciao");  
    StringBuffer s2 = new StringBuffer("Addio");  
    System.out.println(s1);  
    // l'oggetto riferito da s1 non e' ancora  
    // eleggibile per GC  
    s1 = s2;  
    // qui e' eleggibile  
    ...  
}
```



Esempio (3)

[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

Esempio (3)

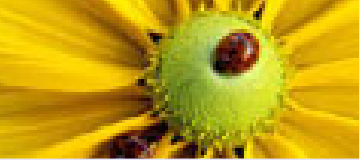
[Esempio \(4\)](#)

[Esempio \(5\)](#)

[Questionario](#)

```
1. import java.util.Date;
2. public class TestGC {
3.     public static void main(String[] args) {
4.         Date d = getDate();
5.         System.out.println(d);
6.     }
7.
8.     public static Date getDate() {
9.         Date d2 = new Date();
10.        String now = d2.toString();
11.        System.out.println(now);
12.        return d2;
13.    }
14. }
```

Tracciare il codice con riferimento alla eleggibilità per GC degli oggetti
(Simulare l'esecuzione e indicare quali oggetti sono eleggibili ad ogni passo)



Esempio (4)

[String e
StringBuffer](#)

[Garbage collection](#)

[Definizioni](#)

[Algoritmo di GC](#)

[Esempio \(2\)](#)

[Esempio \(3\)](#)

[Esempio \(4\)](#)

[Esempio \(5\)](#)

[Questionario](#)

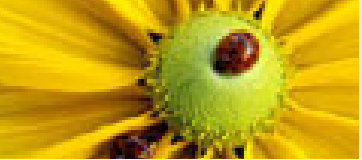
```
public class Isola {
    Isola i;

    public static void main(String[] args) {
        Isola i1 = new Isola();
        Isola i2 = new Isola();
        Isola i3 = new Isola();

        i1.i = i2;
        i2.i = i3;
        i3.i = i1;

        i1 = null;
        i2 = null;
        i3 = null;

        // qui quali oggetti sono eleggibili?
    }
}
```

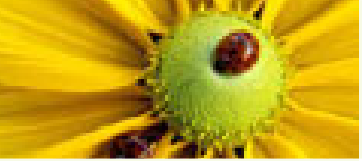



Esempio (5)

Simulare la procedura di GC mark-and-sweep nel punto segnato nel seguente programma:

```
class Employee {
    private String name;
    private Employee boss;
    public final static Employee CEO = new Employee("Gustavo", null);
    ...

    public static void main(String[] args) {
        Employee w = new Employee("Walter", CEO);
        f();
    }
    public static void f() {
        Employee j = new Employee("Jesse", CEO),
            p = new Employee("Pete", j);
        ArrayList<Employee> l = new ArrayList<>();
        l.add(j);
        j = null;
        p = null;
        // simulare la GC a questo punto
    }
}
```



String e
StringBuffer

Garbage collection

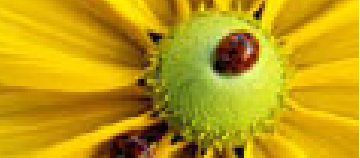
Questionario

D 1

D 2

D 3

Questionario



D 1

String e
StringBuffer

Garbage collection

Questionario

D 1

D 2

D 3

Data la stringa costruita mediante `s = new String("xyzzzy")`, quali delle seguenti invocazioni di metodi modificherà la stringa?

- A. `s.append("aaa");`
 - B. `s.trim();`
 - C. `s.substring(3);`
 - D. `s.replace('z', 'a');`
 - E. `s.concat(s);`
 - F. Nessuna delle precedenti.
-



D 1

String e
StringBuffer

Garbage collection

Questionario

D 1

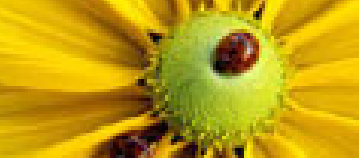
D 2

D 3

Data la stringa costruita mediante `s = new String("xyzzzy")`, quali delle seguenti invocazioni di metodi modificherà la stringa?

- A. `s.append("aaa");`
- B. `s.trim();`
- C. `s.substring(3);`
- D. `s.replace('z', 'a');`
- E. `s.concat(s);`
- F. Nessuna delle precedenti.

F. – Gli oggetti `String` sono immutabili.



D 2

String e
StringBuffer

Garbage collection

Questionario

D 1

D 2

D 3

Qual è l'output del seguente brano di codice?

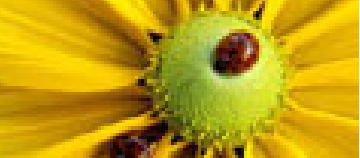
```
1. String s1 = "abc" + "def";  
2. String s2 = new String(s1);  
3. if (s1 == s2)  
4.     System.out.println("A");  
5. if (s1.equals(s2))  
6.     System.out.println("B");
```

A. AB

B. A

C. B

D. Nessun output.



D 2

String e
StringBuffer

Garbage collection

Questionario

D 1

D 2

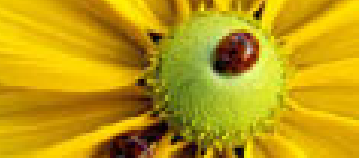
D 3

Qual è l'output del seguente brano di codice?

```
1. String s1 = "abc" + "def";  
2. String s2 = new String(s1);  
3. if (s1 == s2)  
4.     System.out.println("A");  
5. if (s1.equals(s2))  
6.     System.out.println("B");
```

- A. AB
- B. A
- C. B
- D. Nessun output.

C.



D 3

String e
StringBuffer

Garbage collection

Questionario

D 1

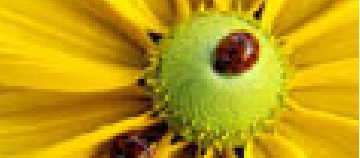
D 2

D 3

Quanti oggetti sono prodotti nel seguente frammento di codice?

```
1. StringBuffer sbuf = new StringBuffer("abcde");  
2. sbuf.insert(3, "xyz");
```

- A.** 1
 - B.** 2
 - C.** 3
 - D.** 4
 - E.** 5
-



D 3

String e
StringBuffer

Garbage collection

Questionario

D 1

D 2

D 3

Quanti oggetti sono prodotti nel seguente frammento di codice?

```
1. StringBuffer sbuf = new StringBuffer("abcde");  
2. sbuf.insert(3, "xyz");
```

A. 1

B. 2

C. 3

D. 4

E. 5

C.