

Cognome

Nome

Matricola

1

Date le classi:

```
public class Padre {
    protected int n;
    protected static int x;
    Padre (int i) {
        n = i;
    }
    public void metodoA (int i) { ...
    }
    public void metodoB (int i) { n=i;
    }
    public static void metodoC (int i){
        x = i
    }
}
```

```
public class Figlio extends Padre {
    Figlio (int i) { ...
    }

    public static void metodoA (int i){ ...
    }
    public void metodoB (int i) { n=i+1;
    }
    public static void metodoC (int i){
        System.out.println (i);
    }
}
```

Implementare il costruttore Figlio(int i) che inizializzi il campo n mediante il parametro i:

Dire se durante la compilazione ci saranno errori e a quale dichiarazione sono dovuti:

Nel caso la compilazione abbia avuto successo, eventualmente eliminando la causa dei problemi, considerando le dichiarazioni seguenti:

```
Padre p = new Padre (3); Figlio f = new Figlio (7);
```

individuare quale metodo viene effettivamente invocato nel codice che segue e cosa si ottiene in output:

```
System.out.println(f.n);
f.metodoB(4);
System.out.println(f.n);
f.metodoC(3);
System.out.println(f.n);
Figlio.metodoC(6);
System.out.println(p.n);
p.metodoB(5);
System.out.println(p.n);
Padre.metodoC(8);
System.out.println(x);
```

output

Scrivere infine l'istruzione necessaria per eseguire il metodoC della classe Padre nel contesto della definizione del metodoC della classe Figlio, nel caso in cui, ad esempio, se ne volessero semplicemente estendere le funzionalità. Rivalutare ora le istruzioni:

```
Figlio.metodoC(7);
System.out.println(x);
Padre.metodoC(8);
System.out.println(x);
```

output

2

Assumendo le dichiarazioni seguenti:

```
final int MAX = 100;
final int MIN = 10;
int i, s=0, valore;
```

Dire qual'è il valore di *i* e *s* dopo aver eseguito:

```
if (valore <= MAX) {i = 2; if (valore > MIN) { i++; s++;}}
```

nei due casi in cui

```
valore == 50
```

```
valore == 5
```

3

Scrivere l'output della seguente istruzione, dopo aver eseguito l'assegnamento numero = 3;

```
if (numero >= 0)
    if (numero == 0) System.out.println("1");
    else System.out.println("2");
    System.out.println("3");
```

4

Assumendo la dichiarazione:

```
Random rand = new Random ();
```

Indicare il range dei valori delle seguenti dichiarazioni:

```
rand.nextInt() % 30;
(int) (Math.random () * 5);
```

Inoltre scrivere un'istruzione per produrre valori pseudo-casuali nell'intervallo:

0 – 10 usando l'oggetto rand

10 – 20 usando il metodo random() della classe Math

5

```
public class While {
    public static void main(String[] args) {
        String originale = "catenella";
        StringBuffer copia = new StringBuffer();
        int i = 0;
        char c = originale.charAt(i);
        while (c != 'e') {
            copia.append(c);
            c = originale.charAt(++i);
        }
        System.out.println(copia);
    }
}
```

```
public class DoWhile {
```

```

public static void main(String[] args) {
    String originale = "catenella";
    StringBuffer copia = new StringBuffer();
    int i = 0;
    char c = originale.charAt(i);
    do {
        copia.append(c);
        c = originale.charAt(++i);
    } while (c != 'e');
    System.out.println(copia);
}

```

scrivere l'output di entrambi i cicli per la stringa originale data e nel caso originale = "edizione"

classe while:

classe DoWhile:

classe while:

classe DoWhile:

6

Nel contesto delle seguenti dichiarazioni:

```

public class Parole {
    final char TAPPO = 0;           // TAPPO è null e indica fine stringa
    private String dato;

    Parole (String s){
        dato = new String (s) + TAPPO;
    }
    int lung () { ...
    }
    String rovescia (int i) { ...
    }
}

```

scrivere il metodo `rovescia`, che riporta la stringa rovesciata, in modo ricorsivo e il metodo `lung`, che calcola la lunghezza della stringa, in modo iterativo.

Suggerimento per implementare il metodo `rovescia`: se il carattere corrispondente all'indice corrente `i` è il TAPPO, allora riporta la stringa vuota altrimenti riporta la stringa ottenuta richiamando ricorsivamente `rovescia` e concatenandola con il carattere corrente. Il metodo `rovescia` verrà chiamato passandogli 0 come parametro.

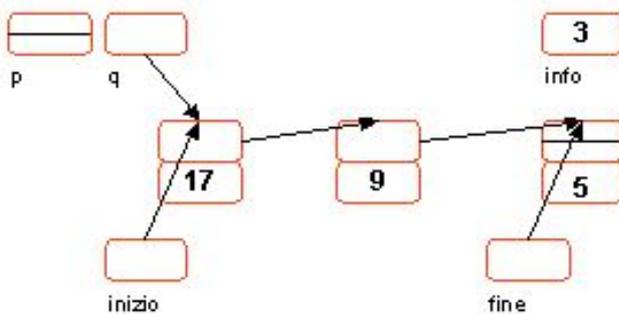
7

Date la classe seguente, che realizza una struttura FIFO (*First In First Out*):

```
public class Lista {
    private Nodo inizio, fine;
    private static class Nodo {
        int info;
        Nodo link;
    }
    Lista () {
        inizio = fine = null;
    }
    void aggiungi (int n) {
        Nodo tmp = new Nodo ();
        tmp.info = n;
        tmp.link = null;
        if (inizio == null)
            inizio = fine = tmp;
        else {
            fine.link = tmp;
            fine = tmp;
        }
    }
    ...
}
```

Implementare il metodo `int rimuovi()` che riporta il primo elemento eliminabile dalla lista.

8



Si usi la classe `Lista` dell'esercizio 7, per definire una lista ordinata. Si scriva il metodo `inserisci(int n)` per effettuare l'inserimento in ordine decrescente nella lista di un nuovo nodo contenente il numero passato come parametro.

Nella figura si suggerisce l'uso di due indici `p` e `q` che vengono inizializzati come visualizzato, per cercare la posizione corretta in cui inserire il nuovo nodo.