

Cognome

Nome

Matricola

1

Data la classe `Vecchia` e la sua sottoclasse `Nuova`:

<pre>public class Vecchia { private int n; protected int getn() { return n; } public int metodo_1(int i) { return n+i; } protected void metodo_2(int i) { n=i+2; } public static void metodo_3(int i) { stampa(i+3); } public static void main(...) { Vecchia v = new Vecchia(); stampa(v.metodo_1(1)); v.metodo_2(2); stampa(v.getn()); Vecchia.metodo_3(9); } }</pre>	<pre>class Nuova extends Vecchia { static int n=20; int m; Nuova() { n--; m=1; } public int getm() { return m; } public int metodo_1(int i) { return m=super.metodo_1(i)+i; } public void metodo_2(double i) { m=(int)i*2; } public static void metodo_3(float i){ } public static void metodo_4 (int i, int j) { stampa(n+i+j); n-- } }</pre>
---	--

Per ciascun metodo della sottoclasse `Nuova` dire se la definizione causa *errore* in compilazione e nel caso, specificare quale, o dire se si tratta di *sovrascrittura* o di *sovraccaricamento* o di *specializzazione*.

metodo_1: **sovrascrittura**

metodo_2: **sovraccaricamento**

metodo_3: **sovraccaricamento**

metodo_4: **specializzazione**

Nel contesto della classe `Nuova`, dopo aver eseguito: `Nuova nv = new Nuova();` valutare le istruzioni:

<code>Vecchia.main(null);</code>	1	<code>stampa(Nuova.n);</code>	18
	4	<code>nv.metodo_2(7.5);</code>	
	12	<code>stampa(nv.getm());</code>	14
<code>stampa(nv.metodo_1(1));</code>	2	<code>nv.metodo_2(7);</code>	
<code>stampa(nv.getn());</code>	1	<code>Nuova.metodo_3(3);</code>	6
<code>Nuova.metodo_4(2,2);</code>	23	<code>stampa(nv.getn());</code>	9

2

Date le classi Padre e Figlio:

<pre>class Padre { int n; public Padre (int i) { n = i; } } class Figlio extends Padre { Figlio() { super(); } }</pre>	<pre>1. public class Test { 2. public void metodo() { 3. Padre p = new Padre(5); 4. Figlio f = new Figlio(10); 5. } 6. public static void main (String[] a) { 7. Test t = new Test(); 8. t.metodo(); 9. } 10. }</pre>
--	---

La classe Test viene compilata correttamente? **NO** Indicare in caso contrario la riga che causa errore: **4**
 Il primo errore che viene segnalato è **Figlio(int) non definito**. Ma risolto questo errore se ne segnala un altro: **Padre() non definito** e perciò non possiamo chiamare `super()` nel costruttore `Figlio()`. Quindi le modifiche necessarie alle classi Padre e Figlio sono due:

Definire un costruttore Figlio(int)

```
Figlio(int i) {
    super(i);
}
```

Definire un costruttore Padre()

```
Padre() {
    n = 0;
}
```

3Scrivere un programma per computare la funzione di Ackermann $A(m, n)$ definita come segue:

$$A(m, n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1, 1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m-1, A(m, n-1)) & \text{se } m > 0 \text{ e } n > 0 \end{cases}$$

```
public static int funzAck (int m, int n) {
    if (m==0) return n+1;
    else
        if (n==0)
            return funzAck(m-1, 1);
        else
            return funzAck(m-1, funzAck(m, n-1));
}
```

Calcolare $A(1, 2)$: **4**Dire il numero di chiamate ricorsive effettuate: **6****4**

Date le due versioni seguenti per il calcolo della potenza:

<pre>static int pot (int k, int n) { if (n == 0) return 1; else return k * pot(k, n-1); }</pre>	<pre>static int pot (int k, int n) { if (n == 0) return 1; else { int t = pot (k, n/2); if ((n % 2) == 0) return t * t; else return k * t * t; } }</pre>
---	--

Produrre una tabella che mostra quante moltiplicazioni e quante chiamate al metodo pot vengono effettuate in ciascuna versione nel calcolare:

8

In una implementazione dinamica della struttura dati stack, definita come segue, dare la definizione della funzione `top()` che accede al valore del nodo in cima allo stack e lo riporta all'ambiente senza modificare la struttura e di un'eccezione non controllata `EmptyStackException()`.

```
public class EmptyStackException extends RuntimeException {
}

public Object top() {
    if (cima == null)
        throw new EmptyStackException();
    else
        return cima.dato;
}
```

<pre>public class Stack { private NodoStack cima; private class NodoStack { Object dato; NodoStack pros; } public Stack() { cima = null; } }</pre>	<pre>public void push(Object o) { NodoStack t = new NodoStack(); t.dato = o; t.pros = cima; cima = t; }</pre>
---	---

Si scriva una istruzione `try-catch` in cui si cattura l'eventuale eccezione lanciata dal metodo `top()` nel contesto di un programma in cui sia stata introdotta la variabile `pila`, onde poter recuperare e inserire almeno un nodo. Gli oggetti da inserire nella pila siano di classe `Integer`.

```
Stack pila = new Stack();
int i = x;
```

```
try {
    if (((Integer)pila.top()).intValue()==x)
        pila.pop();
} catch (EmptyStackException e) {
    pila.push(Integer(x));
}
```

oppure una variante del metodo per la ricerca della palindroma:

```
try {
    while (i>=0 && ((Character)pila.top()).charValue()==pila.charAt(i)) {
        pila.pop(); i--;
    }
} catch (EmptyStackException e) {
    flag=false;
}
```

9

Avendo definito `class Settimana implements java.util.Iterator { }`; dire se sono lecite le seguenti istruzioni:

- Settimana s = new Settimana();
while(s.hasNext()) <computa(s.next())> **SI** NO
- Iterator i = new Settimana(); **SI** NO
- Qual'è il concetto della programmazione a oggetti per cui diversi oggetti che hanno una interfaccia comune rispondono in modo diverso quando un metodo di quell'interfaccia è invocato:
polimorfismo
- Casting è appropriato quando si riceve un riferimento a una classe antenata e si sappia che l'oggetto è una particolare sotto-classe e si voglia usare l'oggetto nella sua funzionalità completa. **SI** NO