Corso di laurea in Comunicazione digitale Fondamenti di architettura e programmazione

AA 2008/09 Prof. M.A. Alberti

IV Appello - 2 luglio 2009

Cognome **Nome**

Matricola

1 (4 punti)

Scrivete il programma Java, **`Esercizio`**, che riceve gli argomenti da riga di commando, li stampa a caratteri maiuscoli, li conta (i token sono separati da spazi bianchi) e stampa il numero ottenuto. Ad esempio:

```
> java Esercizio prova di esame!
```

Produce in output:

```
PROVA DI ESAME!
3
```

---

2 (4 punti)

Supponete di aver istanziato l'oggetto **`tokenizer`** di classe **`StringTokenizer`** con la stringa **`frase`** di input:

```
StringTokenizer tokenizer = new StringTokenizer(frase);
```

Ora vogliamo manipolare con una data regola i singoli token per produrre una traduzione della frase, ad esempio con il ciclo:

```
for (int i=0; i < tokenizer.countTokens(); i++)
    traduzione += traduci(tokenizer.nextToken());
```

Cosa succede con la stringa di input frase "prova di esame di luglio"? Se c'e' un problema spiegate e correggete. Ricordo che il metodo **`countTokens`** calcola il numero di volte in cui può essere chiamato il metodo **`nextToken`** su questo tokenizer prima di generare un'eccezione.

3 (3 punti)

Commentate lo spezzone di codice seguente, spiegando concisamente il suo effetto:

```
String[] archivio = new String[MAX];
for (inti=0; i<MAX; i++)
    out.println(archivio[i].length());
```

4 (3 punti)

Fornite una concisa spiegazione del perchè **`nullPointerException`** NON è un'eccezione controllata.

5 (3 punti)

Stabilite qual'è l'output del seguente codice Java, supponendo che sia stata dichiarata la variabile `int i`:

`for (i = 1; i < 10; i++) {`
`  if (i%3 == 0) break;`
`  System.out.println (i);`
`}`
`System.out.println (i);`	

`for (i = 1; i < 10; i++) {`
`  if (i%3 == 0) continue;`
`  System.out.println (i);`
`}`
`System.out.println (i);`	

`for (i = 1; i < 10; i++) {`
`  if (i%3 == 0) System.exit(0);`
`  System.out.println (i);`
`}`
`System.out.println (i);`	

6 (2 punti)

Supponete che sia definita una classe `BarcaVela` estensione della classe `Barca` e supponete che quest'ultima classe abbia definito i metodi pubblici `velocitàNodi()` e `accendiMotori()`.

Un oggetto di classe `BarcaVela` risponde ai metodi `velocitàNodi()` e `accendiMotori()`? SI NO

Questi metodi dovranno eseguire le stesse azioni sia per gli oggetti di classe `BarcaVela` sia per gli oggetti di classe `Barca`, sempre e comunque? SI NO

Come si chiama in Java il concetto di specializzazione dei metodi?

Se per la classe `BarcaVela` è stato definito il metodo `alzareRanda()`, questo può essere invocato su un oggetto di classe `Barca`? SI NO

---

7 (5 punti)

Data la classe seguente con i costruttori elencati:

```
class Dipendente {
   private String nome;
```

```
Dipendente(){
     nome = new String();
   }
Dipendente(String unNome) {
    nome = unNome;
   }
}
```

E' possibile al di fuori della classe cambiare il valore del membro `nome` di un oggetto già istanziato? SI NO

Come? (scrivere le istruzioni necessarie)

Per estendere la classe con una specializzazione, introduciamo la sottoclasse `Impiegato`, completare quindi il codice seguente opportunamente:

```
class Impiegato extends Dipendente {
   private int settore;
   // 1, 2, 3 ad esempio

   Impiegato() {


    settore = 1;
     // valore di default


   }
```

```
Impiegato (String unNome,
int unSettore) {



    settore = unSettore;




}
}
```

Supponendo di aver eseguito l'istruzione: `Impiegato imp = new Impiegato();`

E' possibile accedere al campo `nome` della superclasse? SI NO

Per accedere al campo `nome` è corretta l'espressione: `imp.nome` ? SI NO

Esistono altri modi? SI NO

Definire il codice di possibili alternative, per accedere al campo:

In generale, in una sottoclasse è possibile accedere direttamente ad un campo privato di un'istanza della superclasse? SI NO

In una sottoclasse è possibile usare un metodo privato della superclasse? SI NO

---

8 (5 punti)

Studiate la seguente funzione ricorsiva di una data classe:

```
public static int f(int m, int n) {
   if (m < 5)
      return n;                                    //indirizzo A
   else if (m > n)
           return 1 + f(m-1, n+1);                 //indirizzo B
         else
           return 1 - f(m-2, n+2);                 //indirizzo C
}
```

calcolate il valore riportato dalla funzione e date una traccia del processo ricorsivo indicando il numero delle chiamate successive alla prima e mettendo in evidenza l'indirizzo in cui avviene la chiamata ricorsiva

```
f(8, 2) = ?
numero delle chiamate (oltre alla prima):
traccia dell'esecuzione:
```

```
f(5, 3) =
numero delle chiamate (oltre alla prima):
traccia dell'esecuzione:
```

```
f(9, 1) =
numero delle chiamate (oltre alla prima):
traccia dell'esecuzione:
```