Corso di laurea in Informatica AA 2006/07 Programmazione, M.A. Alberti

II Compitino, 22 dicembre 2006

Cognome **Nome**

Matricola

1

a. Scrivere la dichiarazione di un array `archivio` di dimensione `DIM` di tipi interi

b. Istanziare l'array `archivio`

c. inizializare l'array `archivio` con i valore crescenti `1, 2 … DIM` mediante un ciclo-for

d. Con la dichiarazione di tipo e l'inizializzazione seguente:

```
String[] unArray = new String[5];
```

Il seguente ciclo-for causa errore o produce un output?

```
for (int i=0; i<unArray.length; i++)  System.out.println(unArray[i]);
```

2

Data la seguente dichiarazione con inizializzazione:

```
String[] zoo =
         {"elefante", "leone", "tigre", "foca"};
```

Dire qual'è l'indice della stringa `leone`.	
Scrivere un'espressione che riporti la stringa dell'ultima posizione dell'array `zoo`.	
Scrivere un'espressione per selezionare il carattere '`g`' in `tigre`.	
Scrivere un'espressione per riportare l'indice di un'occorrenza del carattere '`g`' in `tigre`	
Dire qual è l'indice dell'ultimo elemento nell'array.	
Scrivere il valore di `zoo.length`.	

3

Cercare nell'array zoo la stringa contenuta nella variabile `parola` e riportarne l'indice della posizione in cui si trova o `-1` se non si trova nell'array.

4
Data la seguente interfaccia:

```
public interface Viventi {
   public String comunica();
   public String muoversi();
   public String nutrirsi();
}
```

e la gerachia di classi:

```
public class Essere {
   String nome;
   static int popolazione=0;
   Essere (String n) {
      nome = n; popolazione++;
   }
   public String comunica() { return "non comunica esplicitamente"; }
   public String toString() { return "nome: " + nome; }
}

public ..... class Mammiferi extends Essere implements Viventi {
   protected String famiglia;
   protected int arti;
   Mammiferi(String n, String f, int a){




   }
   public String toString() {
      return super.toString() + " della famiglia " + famiglia; }
   public String comunica() { return "comunica a suoni e gesti"; }
   public abstract String muoversi();
   public abstract String nutrirsi();
}

public class Ominidi extends Mammiferi implements Viventi{
   String lingua;
   String cibo;
   Ominidi(String n, String f, String ln, String c){
      super(n, f, 2); lingua = ln; cibo = c;
   }
   public String comunica() { return "parla " + lingua; }
   public String muoversi() { return "cammina e corre"; }
   public String nutrirsi() { return "preferisce "+cibo; }
}
```

a. completare la dichiarazione della classe `Mammiferi`
b. completare il costruttore della classe `Mammiferi` utilizzando i parametri passati per inizializzare i campi propri e quelli ereditati correttamente
c. commentare l'istruzione `new Mammiferi(nome, famiglia, arti);` dove `nome`, `famiglia` e `arti` sono opportune variabili d'ambiente
d. dire in che relazione sta il metodo `comunica()` della classe `Mammiferi` con il metodo `comunica()` della classe `Esseri`

e. in `Ominidi` è possibile definire un altro metodo `comunica(String p)`? `SI NO`

f. nel caso trattato in e. si dice che si effettua un ……

5

Implementare la sottoclasse `Canidi` di `Mammiferi`, che dichiari un campo `cibo` contenente sotto forma di stringa l'indicazione del cibo preferito. Inoltre la classe deve sovrascrivere il metodo `comunica()` e deve implementare i metodi astratti della classe `Mammiferi`, richiesti dalla interfaccia `Viventi`.

Prima di iscrivere il codici si studi l'effetto della dichiarazione e inizializzazione:
`Canidi c=new Canidi("fido", "Setter", "pezzato", "carne");`
e le chiamate dei metodi seguenti con il loro valore computato o stampato:

```
out.println(c)        nome: fido della famiglia Setter
c.comunica()               comunica a suoni e gesti, abbaia
c.muoversi()               si muove a 4 zampe
c.nutrirsi()               si nutre di carne
```

```
public class Canidi …… …… ……{

   String cibo;
   Canidi(String n, String f, String m, String c){




   }
   public String comunica() {                                    }

   public String nutrirsi() {                                    }

   public String muoversi() {                                    }

}
```

a. Dichiarare una classe `Mondo` con un campo `micromondo` definito come array di `Essere`. Il costruttore di un oggetto `Mondo` dovrà inizializzare l'array alla dimensione passatagli come parametro.

b. È possibile eseguire le istruzioni: `Mondo m = Mondo(5);` SI NO

```
m.micromondo[0]=new Essere("sasso");
m.micromondo[1]=new Essere("fiume");
m.micromondo[2]=new Canidi("fido", "Setter", "pezzato", "carne");
m.micromondo[3]=new
      Ominidi("Andrea","Neanderthal","inglese","fish&chips");
m.micromondo[4]=new
      Ominidi("Lucy","Australopiteco","arabo","cuscus");
```

c. È corretto scrivere `m.micromondo[i].comunica();` SI NO

d. Quale metodo comunica() viene effettivamente chiamato?

e. Volendo eseguire il metodo `nutrirsi` per il terzo elemento dell'array `micromondo` occorre scrivere:

f. È corretto scrivere l'espressione `Essere.popolazione`? SI NO

g. Se si come viene valutata:

6

Supponendo di avere definito una classe `Rectangle` con due campi `width` e `height`, dire se è corretto il codice:

```
public class TestRectangle {
     public static void main(String[] args) {
          Rectangle unRettangolo;
          unRettangolo.width = 40;
          unRettangolo.height = 50;
     }
}
```

SI NO

7

C'è un problema con la dichiarazione di interfaccia seguente ? SI NO

```
public interface BuoneManiere {
     public void saluto (String par) {
          System.out.println (par);
      }
}
```

8

Se il codice precedente è da corregere, modificatelo in modo che sia corretto.

9

Scrivete un metodo **rovescia** che accetta un parametro di tipo String e riporta un dato di tipo String che contenga i caratteri del parametro in ordine inverso, realizzato mediante un ciclo-for. Ad esempio rovescia ("`adamo`") produce la stringa "`omada`".