Corso di laurea in Comunicazione digitale Fondamenti di architettura e programmazione

AA 2008/09 Prof. M.A. Alberti

III Appello - 4 giugno 2009

Cognome **Nome**

Matricola

1 (4 punti)

Scrivete il programma Java, Esercizio, che produce in output in ordine inverso gli argomenti che gli vengono passati sulla riga di commando. Ad esempio:

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

Produce in output:

```
esame!
di
prova
```

```
public class Esercizio {

    public static void main(String[] args) {
        for (int i=args.length-1; i>=0; i--)
            System.out.println(args[i]);
    }
}
```

---

2 (3 punti)

Supponete di aver istanziato l'oggetto **v** di classe **Vector** di lunghezza 3, con valori di tipo **String** e che l'output dell'istruzione **println(v)** sia:

```
[oggi, domani, mai]
```

Considerate ora il seguente ciclo, **scorretto**, che dovrebbe premettere a ogni elemento del vettore la stringa **non**

```
for (int i=0; i < v.size(); i++) v.add(i, "non");
```

Se il ciclo funzionasse correttamente l'istruzione **println(v)** dovrebbe dare:

```
[non, oggi, non, domani, non, mai]
```

Putroppo non è così. Individuate il problema e scrivete il codice Java necessario per correggere il ciclo.

il problema nasce perché ad ogni passo la dimensione del vettore è aumentata dal nuovo inserimento. Quindi occorre archiviare la dimensione originale in una variabile, ad esempio **int ln = v.size();** e avanzare l'indice di due posizioni per tener conto del nuovo inserimento

```
public class Vettori {

    public static void main(String[] args) {
        Vector<String> v = new Vector<String>();
        v.add("oggi"); v.add("domani"); v.add("mai");
```

```
        int ln = v.size();
        for (int count=0, pos=0; count<ln; count++, pos=pos+2)
            v.add(pos, "NON");
        System.out.println(v);
    }
}
```

---

3 (3 punti)

Fornite un piccolo esempio concreto di come un metodo può causare l'eccezione **nullPointerException.**

```
public class NullPointerExcClass {

    String[] field;
    NullPointerExcClass(int l){
        field = new String[l];
    }

    public static void main(String[] args) {
        final int MAX = 10;
        NullPointerExcClass test = new NullPointerExcClass(MAX);

        /* lancia un'eccezione perche' l'elemento stringa non e'
         * stato instanziato e quindi non puo' eseguire il metodo
         * length() o qualunque altro. */
        for (int i=0; i<MAX; i++){
            System.out.println(test.field[i].length());
        }
    }
}
```

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

Solo a run time si può controllare se un oggetto è stato creato. Staticamente il compilatore può solo controllare che il metodo che viene richiamato sia tra quelli disponibili per un dato oggetto, ma non può sapere se l'oggetto è stato creato.

---

4 (5 punti)

Considerate il seguente spezzone di programma che chiede all'utente di immettere 2 numeri interi e ne calcola la divisione.

```
ConsoleInputManager in = new ConsoleInputManager();
int n1, n2;
double r;
n1 = in.readInt("input un intero: ");
n2 = in.readInt("input un secondo intero: ");
r = (double) n1/n2;
```

Inserite il codice in una istruzione **try-catch** con più clausole **catch** in modo da controllare gli errori seguenti:

- tentativo di divisione per 0
- introduzione di dati testuali invece di interi (**InputMismatchException**)

Se queste condizioni si verificano il programma lo segnala all'utente e riprende l'esecuzione con la richiesta di immissione dei dati

```
public class TryCatch {


    public static void main(String[] args) {
        ConsoleInputManager in = new ConsoleInputManager();
        ConsoleOutputManager out = new ConsoleOutputManager();
        int n1, n2;
        double r;
        boolean notOk = true;
        while (notOk) {
            try {
                n1 = in.readInt("Un intero: ");
                n2 = in.readInt("Un secondo intero: ");
                r = n1/n2;
                out.println(r);
                notOk = false;
            }
            catch (ArithmeticException e) {
                out.println("Tentativo di divisione per 0");
            } catch (InputMismatchException e) {
                out.println("Input diverso da intero");
            }
        }
    }
}
```

---

5 (6 punti)

Le istruzioni di un programma si sono inspiegabilmente mescolate. Dovete ricostruire l'ordine preciso delle istruzioni perché in esecuzione dia il seguente risultato:

```
> Java Puzzle Olio
strippa
> Java Puzzle Stanlio
strappa
```

Ricostruite le righe d'istruzioni nell'ordine corretto. Notate che potrebbero mancare delle parentesi e dei punti e virgola.

```
try    {
System.out.print(“s”);          System.out.print(“a”);
        System.out.print(“i”);
}    finally   {
catch (MiaEccezione e)  {
                                    System.out.print(“a”);
System.out.print(“pp”);
throw new MiaEccezione();
        public static void main(String[] arg) {
            String test = arg[0];
rischia(test);
        if “Olio”.equals(arg))
                static void rischia(String arg) throws MiaEccezione
                {
                    System.out.print(“tr”);
class MiaEccezione extends Exception{}
                                            System.out.println()
public class Puzzle {
```

Ordine corretto:

```
class MiaEccezione extends Exception{}

public class Puzzle {

    static void rischia(String arg) throws MiaEccezione{
        System.out.print("tr");
        if (arg.equals("Olio")) {
            System.out.print("i");
            throw new MiaEccezione();
        }
        System.out.print("a");
    }

    public static void main(String[] args) {
        String test = args[0];
        System.out.print("s");
        try{
            rischia(test);
        }catch (MiaEccezione e){
        }finally {
            System.out.print("pp");
            System.out.print("a");
            System.out.println();
        }
    }
}
```

---

6 (10 punti)

Nella classe **Morse** definite un campo statico di identificatore **morseCode** come array di lunghezza 26 di stringhe:

```
public class Morse {
    static String[] morseCode = new String[26];
}
```

Il metodo statico **init()** lo inizializzerà secondo la convenzione dell'alfabeto Morse, vedi punto 7 successivo.

Scrivere il metodo statico **String code(String parola)** che codifica i caratteri che compongono la parola passata come argomento secondo l'alfabeto Morse e riportano all'ambiente chiamante la stringa di caratteri codificata. Fate uso della classe **StringBuffer** che offre il metodo **append(String s)** utile allo scopo.

```
public static String code(String word){
        StringBuffer codeStr = new StringBuffer();
        for (int i=0; i<word.length(); i++)
            codeStr.append(morseCode[word.charAt(i)-'a']);
        return codeStr.toString();

}
```