Esame di Programmazione II, 15 luglio 2013 (2 ore)

Un foglio elettronico è una tabella bidimensionale di celle. Dentro le celle è posizionata un'espressione, il cui valore è visualizzato dentro la cella. L'espressione di una cella può essere una costante numerica intera, una stringa, un'operazione aritmetica fra due altre espressioni, la concatenazione fra due altre espressioni o un riferimento a una cella. Per esempio, il seguente programma costruisce un foglio eletronico, popola alcune sue celle con espressioni e infine lo stampa:

```
public class Main {
  public static void main(String[] args) {
    Sheet sheet = new Sheet(5, 6);
    sheet.setCell(1, 0, new NumericConstant(13));
    sheet.setCell(2, 0, new NumericConstant(-45));
    sheet.setCell(1, 1, new NumericConstant(11));
    sheet.setCell(2, 1, new NumericConstant(23));
    sheet.setCell(1, 2, new NumericConstant(1));
    sheet.setCell(2, 2, new NumericConstant(4));
    sheet.setCell(0, 3, new StringConstant("totale:"));
    sheet.setCell(1, 3, new Add(sheet.getCell(1, 0), new Add(sheet.getCell(1, 1), sheet.getCell(1, 2))));
    sheet.setCell(2, 3, new Add(sheet.getCell(2, 0), new Add(sheet.getCell(2, 1), sheet.getCell(2, 2))));
    sheet.setCell(1, 4, new Div(sheet.getCell(2, 3), new NumericConstant(0)));
    sheet.setCell(2, 4, new Append(sheet.getCell(2, 3), sheet.getCell(1, 4)));
    sheet.setCell(3, 4, new Append(sheet.getCell(2, 3), sheet.getCell(0, 3)));
    sheet.setCell(3, 5, new Add(sheet.getCell(0, 1), sheet.getCell(3, 4)));
    //sheet.setCell(1, 0, sheet.getCell(1, 3));
    System.out.println(sheet);
  }
}
```

ottenendo a video:

```
|         |       13|      -45|         |         |
|         |       11|       23|         |         |
|         |        1|        4|         |         |
|   totale:|       25|      -18|         |         |
|         |      ###|      ###|-18 totale|         |
|         |         |         |      ###|         |
```

dove `###` indica un errore di valutazione. Se si decommenta il penultimo comando del `main`, si crea un ciclo nel foglio elettronico e la sua stampa risulta differente:

```
|         |      @@@|      -45|         |         |
|         |       11|       23|         |         |
|         |        1|        4|         |         |
|   totale:|      @@@|      -18|         |         |
|         |      ###|      ###|-18 totale|         |
|         |         |         |      ###|         |
```

Esercizio 1 [2 punti] Si definisca l'eccezione `EvaluationException` che estende `java.lang.RuntimeException` ed indica che la valutazione di un'espressione è fallita per un errore di tipo o una divisione per zero. Si definisca la sua sottoclasse `CyclicEvaluationException` che indica che la valutazione di un'espressione è finita in un ciclo.

Esercizio 2 [3 punti] Si completi la seguente classe, che implementa un'espressione posta in una cella:

```
public abstract class Exp {
  public abstract int getNumericValue() throws EvaluationException;
  public abstract String getStringValue() throws EvaluationException;
  @Override public final String toString() { ... }
}
```

Il metodo `getNumericValue()` restituisce il valore dell'espressione se tale valore è numerico; lancia un'eccezione fra quelle dell'esercizio 1 altrimenti. Il metodo `getStringValue()` restituisce il valore dell'espressione visto come stringa; questo è possibile sempre, anche se l'espressione ha un valore intero, poiché un intero può essere convertito nella stringa corrispondente. Il metodo `toString()` si comporta come un sinonimo di

`getStringValue()` ma non lancia eccezioni. Infatti, se la valutazione dà un errore, deve restituire `###` oppure `@@@`, a secondo della classe dell'eccezione.

Esercizio 3 [**8 punti**] Si completino le seguenti sottoclassi concrete di `Exp`, implementando quindi i suoi metodi astratti. Non ci si preoccupi per ora di controllare se la valutazione di un'espressione finisce in un ciclo.

```
public class NumericConstant extends Exp {  // costante numerica
  public NumericConstant(int value) { ... }
  {...}
}

public class StringConstant extends Exp {  // costante stringa
  public StringConstant(String value) { ... }
  {...}
}

public class Add extends Exp {  // addizione di due espressioni
  public Add(Exp left, Exp right) { ... }
  {...}
}

public class Div extends Exp {  // divisione fra due espressioni
  public Div(Exp left, Exp right) { ... }
  { ... attenzione al caso in cui il valore numerico di right e' zero! }
}

public class Append extends Exp {  // concatenazione fra due espressioni viste come stringa
  public Append(Exp left, Exp right) { ... }
  { ... }
}

public final class Cell extends Exp {  // un riferimento a una cella
  private Exp exp;
  protected Cell() { this.exp = new StringConstant(""); } // una cella nasce vuota
  public void setExp(Exp exp) { this.exp = exp; }
  { ... }
}
```

Esercizio 4 [**3 punti**] Si completi la classe che implementa il foglio elettronico vero e proprio:

```
public class Sheet {
  private final Cell[][] cells;
  public Sheet(int width, int height) { ... inizializza cells a celle vuote }
  public void setCell(int x, int y, Exp exp) { ... setta exp come espressione della cella in x,y }
  public Cell getCell(int x, int y) { ... ritorna la cella in x,y }
  @Override public String toString() {
    String result = "";
    for (int y = 0; y < cells[0].length; y++) {
      result += "|";
      for (int x = 0; x < cells.length; x++) {
        String content = cells[x][y].toString();
        if (content.length() > 10) content = content.substring(0, 10);
        result += String.format("%10s|", content);
      }
      result += "\n";
    }
    return result;
  }
}
```

Esercizio 5 [**6 punti**] Si modifichino la classe `Exp` e le sue sottoclassi concrete in modo da fare lanciare una `CyclicEvaluationException` ai metodi `getNumericValue()` e `getStringValue()` quando la valutazione finisce in un ciclo e quindi non può terminare.