Exercise 1

On considère la classe abstraite Bloc:

```
public abstract class Bloc implements Comparable<Bloc>{
   protected String couleur;
   protected int hauteur;
   public Bloc(String couleur,int hauteur) {
      this.couleur=couleur;
      this.hauteur=hauteur;
   }
   public int getHauteur() {
      return hauteur;
   }
    public String getCouleur() {
      return couleur;
    }
    public abstract boolean estEmpilable();
    public boolean equals(Object o) {
      ...
    }
    public int compareTo(Bloc b) {
      ...
    }
    public String toString() {
      ...
    }
}
```

Il existe des blocs de différentes formes: Cube, Cylindre et Cone. Les cônes ont la particularité de ne pas être empilables (on ne peut rien poser dessus).

- Donnez le code de la méthode `public boolean equals(Object o)` permettant de comparer deux blocs. Deux blocs sont identiques s'il ont la même couleur et sont de même hauteur.
- Donnez le code de la méthode `public int compareTo(Object o)` permettant de comparer deux blocs. La comparaison a lieu sur la base des hauteurs des deux blocs.
- Donnez le code de la méthode `public String toString()` affichant les informations sur la hauteur, la couleur et l'empilabilité du bloc.
- Donnez un code Java pour les classes Cube, Cylindre et Cone. Hauteur et couleur d'un cube sont fixées à la création.

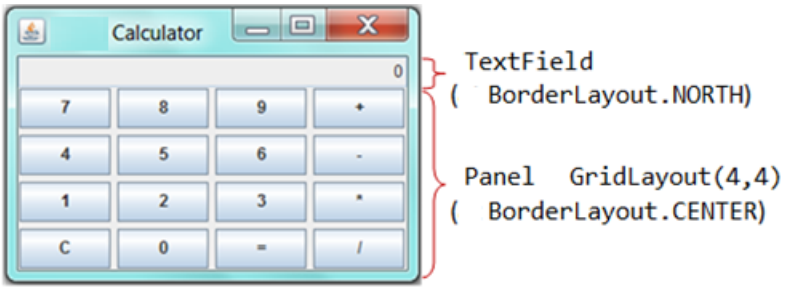
La classe Empilement permet de représenter des empilements de blocs. La pile des blocs est gérée par un attribut de type ArrayList de Bloc. Cette classe dispose des méthodes :

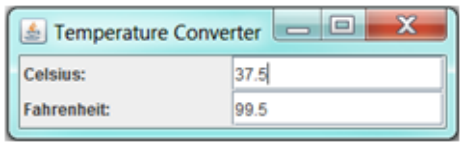
- `public int RecupereHauteur()` qui renvoie la hauteur de l'empilement. Cette hauteur correspond au cumul des hauteurs des blocs qui composent l'empilement.

- `public void ajoute(Bloc b)` qui permet d'ajouter à l'empilement un bloc passé en paramètre. Une exception EmpilementImpossibleException est déclenchée si le bloc n'est pas empilable.
- `public void trier()` qui trie la liste des blocs.
- Donnez un code Java pour cette classe Empilement.
- Donnez le code d'une méthode main d'une classe Test qui
  - Crée 1 cube bleu de hauteur 4, 1 cylindre rouge de hauteur 2 et un cone bleu de hauteur 5,
  - Crée un empilement pile
  - Tente d'ajouter à pile successivement le cube, le cone et le cylindre ci-dessus, si un des ajouts est impossible un messsage est affiché, sans interruption du programme,
  - Affiche la hauteur de l'empilement créé,
  - Proposez une amélioration du programme en déclenchant une exception si un cone est ajouté à l'empilement. L'affichage de la hauteur de l'empilement devra toujours avoir lieu, même si ne tentative d'ajout de cone a été effectué.

Exercise 2

Reprenez Le codes des IHM ci-dessous réalisée lors du TP précédent.





- Rendre La calcuatrice opérationnelle. Le code ci-dessous permet à partir d'une chaine de caractère décrivant une expression mathématique de réaliser cette dernière et de retourner le résultat.

```
import javax.script.ScriptEngineManager;
import javax.script.ScriptEngine;
import javax.script.ScriptException;

public class Test {
  public static void main(String[] args) {
  try{
    ScriptEngineManager mgr = new ScriptEngineManager();
    ScriptEngine engine = mgr.getEngineByName("JavaScript");
    String foo = "40+2/56-36*5";
    System.out.println(engine.eval(foo));
    }catch(ScriptException e){
      System.out.println("Expression mathematique Incorrecte!");
    }
```

```
}
}
```

- Pour le convertisseur celcius/fahrenheit, utiliser l'interface FocusListener.