MODIFIERS

Scenario

A Bowl HAS-A Snake (aggregation). As the temperature of the Bowl increases, the height of the Snake increases. The maximum height the Snake can reach is 120cm. This is a FIXED value and cannot be changed.

If the Snake reaches this maximum height, it means the temperature of the Bowl is very high and the Bowl cracks and the Snake escapes.

Draw UML diagrams showing the relationship between the Bowl and Snake classes. You do not have to include attributes or methods, only the class title. Make sure each diagram has 3 tiers and a line demonstrating the relationship between each class.

```
package bowlsnakepackage

public class Snake{

  private final int maxHeight = 120;

  private int currentHeight;

  public Snake(){

    this.currentHeight = 0;

  }

  public int getCurrentHeight(){

    return this.currentHeight;

  }

  public int getMaxHeight(){

    return this.maxHeight

  }

  public void setCurrentHeight(int newHeight){

    this.currentHeight = newHeight;

  }

}
```

Here is the Snake class:

i. Circle the constructor method

ii. How many public **modifiers** are there?

iii. Underline an **attribute** whose value cannot be changed.

iv. Explain your answer to iii:

Notice that the Snake class is a member of a package. In fact, as you will see, all classes in this project are members of the same package.

Here is the Bowl class:

```
package bowlsnakepackage

public class Bowl{
  private int bowlTemp;
  private Snake snake;
  private boolean isCracked;
  public Bowl(Snake snake){
    this.snake = snake;
    this.bowlTemp = 20;
    this.isCracked = false;
  }
  public int getBowlTemp(){
    return this.bowlTemp;
  }
  public void setBowlTemp(){
    //set the bowl's temp between 20 and 120
    bowlTemp = (int)(Math.random()*101)+20;
    //move the Snake according to the temperature
    //update the bowl's cracked status
    moveSnake();
    checkCracked();
  }
  private void moveSnake(){
    snake.setCurrentHeight(bowlTemp);
  }
  private void checkCracked(){
    if(bowlTemp >= 120){
      this.isCracked = true;
    }
  }
  public boolean getCracked(){
    return this.isCracked;
  }
}
```

i. circle a mutator method

ii. Describe the purpose of a mutator method

ii. how many methods use the private **modifier**?

iii. why do these methods use the private **modifier** and not the public modifier?

The Bowl class is also a member of the bowlsnakepackage package.

Classes in the same package should **trust** each other!

So, instead of making their attributes **private**, they could/should use the **protected** modifier. This means classes ***in the same package*** will be able to see the data in the attributes of ***other package classes***. There will be no need to call accessor/mutator methods.

Classes **outside the package** will still need to use accessor and mutator methods for attributes using the **protected** modifier.

Adapt the `Snake` class by making the 2 attributes **protected**:

Now, adapt the `moveSnake` method of the `Bowl` class. How can we change it to take advantage of the `protected` modifier of `Snake` class attributes. Rewrite the method here:

So far we have become familiar with 2 modifiers: public and private

This unit has introduced 2 more: final and protected.

A variety of modifiers are available to designers of computer programs and the more you become familiar with them, the more effective your computer programs will become.