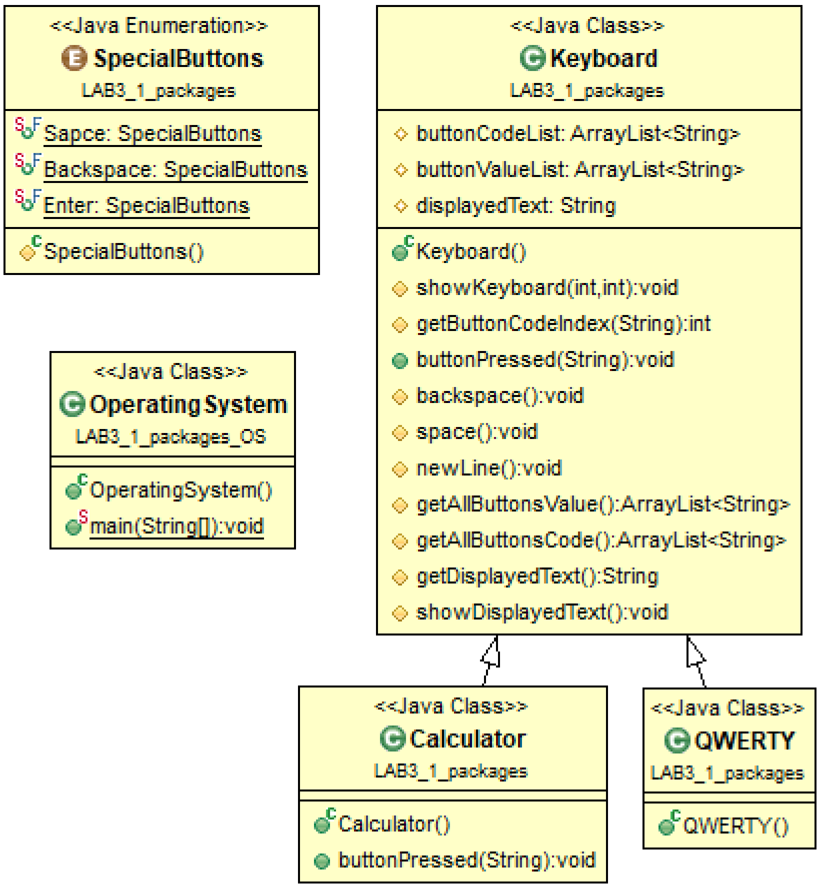
Packages and Polymorphism

Objective:

To use packages in java to encapsulate a group of classes that can grant a strong access control. In addition, packages help making simulation more realistic. Also, in this lab polymorphism is used to get the maximum benefit from the last lab.

Part#1 Adding packages to the keyboard simulator:

In the previous lab, we made the QWERTY keyboard and Calculator keyboard that extends the keyboard class. Fortunately, an Operating System company contacted you and they want to use your keyboards as virtual keyboards in their OS. You made the agreement with them to provide them with the functionality they need, but not the source code. Therefore, you agreed that they can make an object from the QWERTY keyboard and Calculator keyboard – they can also make an object from the keyboard class with no benefit- and use the buttonPressed method inside the keyboard class. While you were discussing how you can access your class, you used the following UML diagram of the classes.



From the UML diagram, they notice that most of the code is in the keyboard class and there is almost no code in the subclasses. This means that they have to use polymorphism to make variable whose type is keyboard and one of the two objects from the other two keyboards. Furthermore, they notice that you put their class in a different package. They didn't mind since the agreement wasn't include sharing the source code.

In general, their code has to start by asking the user of their OS which keyboard they want to use and according to their answer one of the two keyboards is going to be made. Here is the main method code they provided:

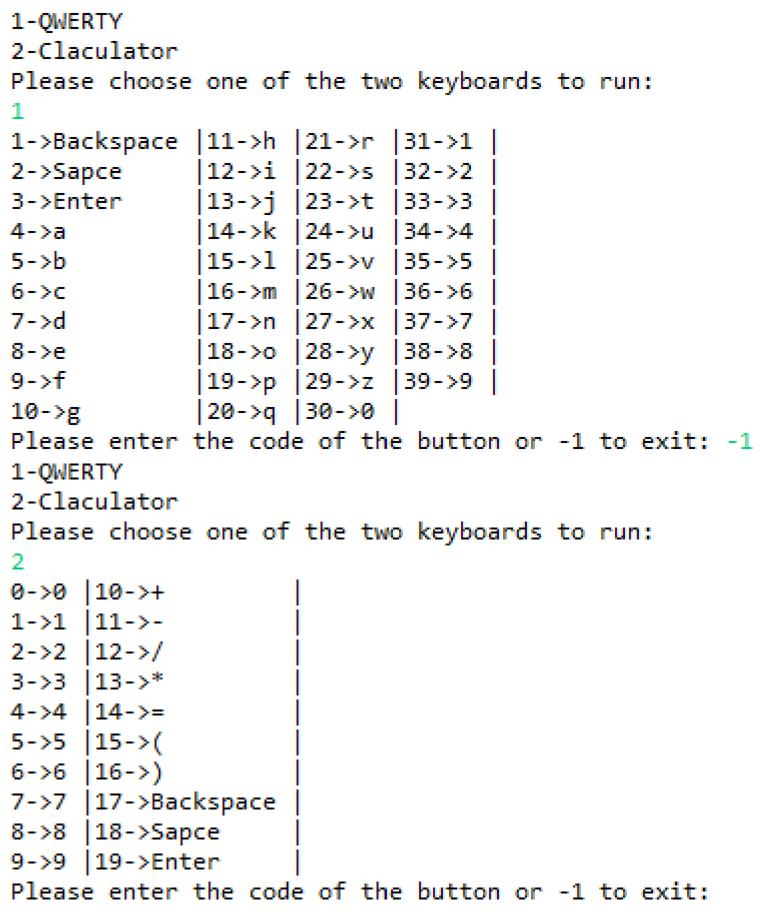
```java
public class OperatingSystem {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    while(true) {
       //ask the user which keyboard to run
       System.out.println("1-QWERTY");
       System.out.println("2-Claculator");
       System.out.println("Please choose one of the two keyboards to run: ");
       int x = input.nextInt();

       //make the keyboard object.
       Keyboard keyboard;
       if(x==1)       keyboard = new QWERTY();
       else           keyboard = new Calculator();

       String inputTxt;
       while(true) {
         System.out.print("Please enter the code of the button or -1 to exit: ");
         inputTxt = input.next();
         if(inputTxt.equals(-1+""))break;
         keyboard.buttonPressed(inputTxt);
       }
    }
  }
}
```

Your task is to make this scenario happen. You will start by copying your code from the last lab, modify it according to the UML, add the OS class in a different package, and do all everything that can make the code run.



```
1-QWERTY
2-Claculator
Please choose one of the two keyboards to run:
1
1->Backspace |11->h |21->r |31->1 |
2->Sapce     |12->i |22->s |32->2 |
3->Enter     |13->j |23->t |33->3 |
4->a         |14->k |24->u |34->4 |
5->b         |15->l |25->v |35->5 |
6->c         |16->m |26->w |36->6 |
7->d         |17->n |27->x |37->7 |
8->e         |18->o |28->y |38->8 |
9->f         |19->p |29->z |39->9 |
10->g        |20->q |30->0 |
Please enter the code of the button or -1 to exit: -1
1-QWERTY
2-Claculator
Please choose one of the two keyboards to run:
2
0->0 |10->+           |
1->1 |11->-           |
2->2 |12->/           |
3->3 |13->*           |
4->4 |14->=           |
5->5 |15->(           |
6->6 |16->)           |
7->7 |17->Backspace |
8->8 |18->Sapce     |
9->9 |19->Enter     |
Please enter the code of the button or -1 to exit:
```

Part#2 Upgrading your source code:

I know that you are happy now because you sold your code which works flawlessly. However, you have more responsibilities. The OS company approached you saying that they want to have more control. They want to give you all the characters they need to be on each keyboard. In way they want to choose the buttons on all your keyboards plus some functional buttons. Here is the code they modified.

```java
public class OperatingSystem {
  //General buttons value that should be in every keyboard
  public enum SpecialButtons { Backspace };

  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    //initiate the values of the buttons
    String[] qwertyButtonsValues = {"a","b","c","d","e","f","g","h","i",
                                    "j","k","l","m","n","o","p","q","r",
                                    "s","t","u","v","w","x","y","z",
                                    "0","1","2","3","4","5","6","7","8","9"};
    String[] calculatorButtonsValues = {"0","1","2","3","4","5","6","7",
                                    "8","9","+","-","/","*","=","(",")"};
    //initiate the values of the general buttons
    String[] generalButtonsValues = {"Space","New Line"};
    String[] generalButtonsActions = {" ","\n"};
    //initiate the values of the buttons
    SpecialButtons[] SpecialButtonsList = {SpecialButtons.Backspace};

    while(true) {
      //ask the user which keyboard to run
      System.out.println("1-QWERTY");
      System.out.println("2-Claculator");
      System.out.println("Please choose one of the two keyboards to run: ");
      int x = input.nextInt();

      //make the keyboard object.
      Keyboard keyboard;
      if(x==1) keyboard = new QWERTY(1,qwertyButtonsValues,generalButtonsValues,
                                        generalButtonsActions,SpecialButtonsList);
      else keyboard =new Calculator(0,calculatorButtonsValues,generalButtonsValues,
                                        generalButtonsActions,SpecialButtonsList);
      String inputTxt;
      while(true) {
        System.out.print("Please enter the code of the button or -1 to exit: ");
        inputTxt = input.next();
        if(inputTxt.equals(-1+""))break;
        keyboard.buttonPressed(inputTxt);
      }
    }
  }
}
```

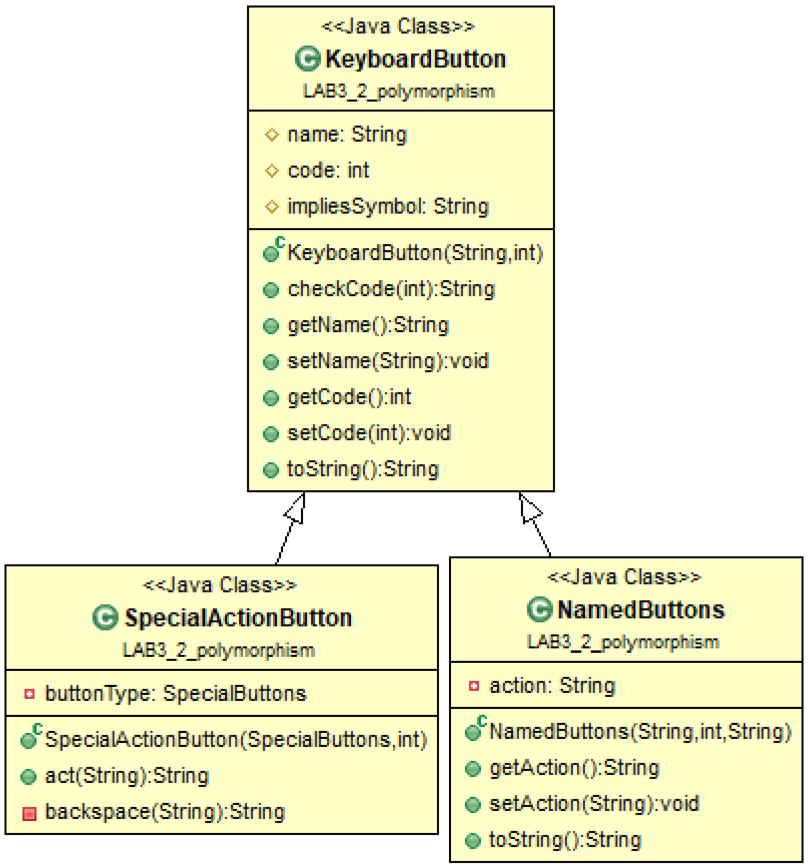
As you can see, they want to have three types of buttons. One for a normal character like the letters and numbers. The other type is the general buttons where they provide the name of the button -that will appear to the user- in the `generalButtonsValues` the array and the button real value of the button in the `generalButtonsActions` array. The final type, is the buttons from the enumeration class which is only the backspace button. They provide it in an array type of SpecialButtons which you agreed on doing some actions they can't provide in the `generalButtonsActions`. Notice the SpecialButtons enumeration class used to be in the keyboard class!

Also, you noticed that they are providing the first code of the first button in each keyboard. For example, the creation of the QWERTY keyboard starts by 1 which is the first code of the first button in the QWERTY keyboard.

You went back to your team and decided on making the constructor of the QWERTY keyboard and Calculator keyboard call their supertype constructor since they are all doing the same. Here is one example:

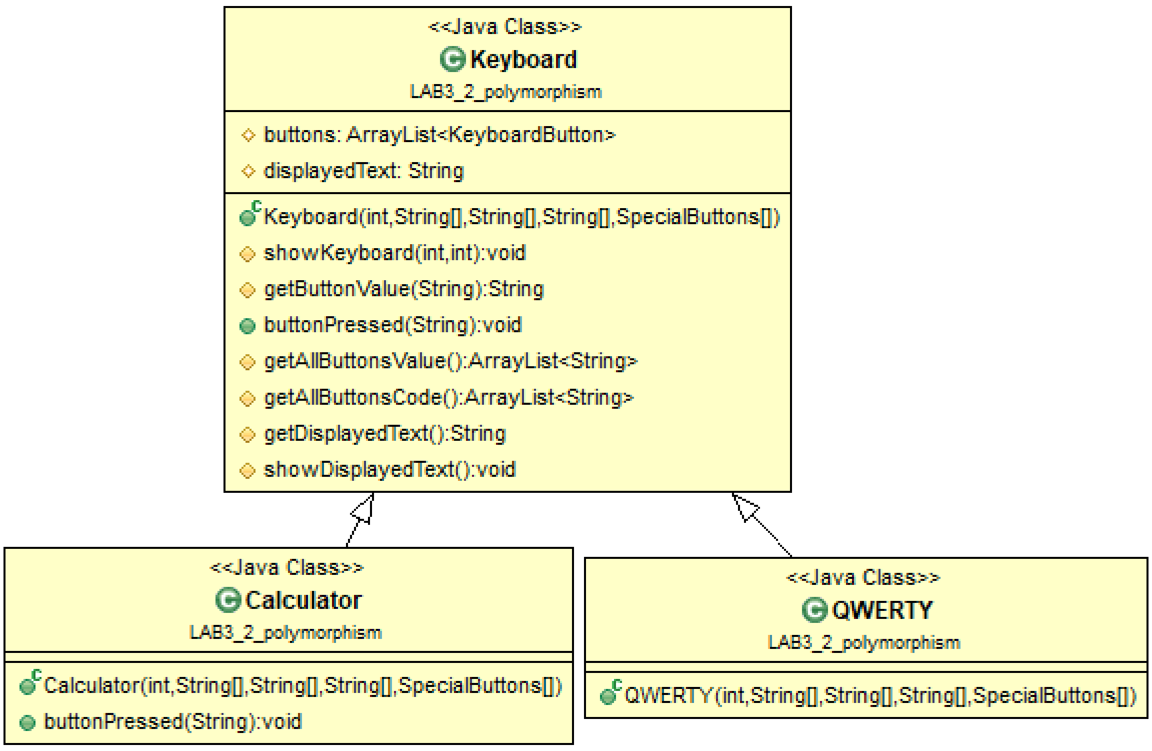
```
public class QWERTY extends Keyboard {
  public QWERTY(int firstCode, String[] keyboardButtonsValues, String[]
  generalButtonsValues, String[] generalButtonsActions, SpecialButtons[]
  SpecialButtonsList) {
      super(firstCode, keyboardButtonsValues, generalButtonsValues,
      generalButtonsActions, SpecialButtonsList);
  }
}
```

Also, you upgraded the keyboard class and took some functionalities from it. You decided that code belong to a button should be taken out of the keyboard class. This means that you have to create another class to hold the buttons data and code. Here is the new set of classes that are related to buttons:



The KeyboardButton class has the name of the button that the user should see, the code of the button, and the implies symbol (->). This class can also check the code and return a string representation of the object (e.g., 1->a). There is a subclass called the NamedButtons which holds the action of the button whose name is different than the action. For example, you can use the (a) letter as the name of the button and the action of the button, but the OS company wants you to show the space button as a word and in the real action is simply an empty character add to the displyedText. The space button is an example of a NamedButton object and the letter (a) is example of the a KeyboardButton object. The last class is for the SpecialActionButton class that has a method called backspace which trims the last character of the displayedText. The only accessible method is the act method that takes a string and match it to the string represntation of the SpecialButton object stored in the object. If they match, then the backspace method is called.

Since all the buttons actions is moved to a new set to classes. The new UML diagram of the Keyboard classes should look like the following:



<<Java Class>>
Keyboard
LAB3_2_polymorphism
buttons: ArrayList<KeyboardButton>
displayedText: String
Keyboard(int,String[],String[],String[],SpecialButtons[])
showKeyboard(int,int):void
getButtonValue(String):String
buttonPressed(String):void
getAllButtonsValue():ArrayList<String>
getAllButtonsCode():ArrayList<String>
getDisplayedText():String
showDisplayedText():void
<<Java Class>>
Calculator
LAB3_2_polymorphism
Calculator(int,String[],String[],String[],SpecialButtons[])
buttonPressed(String):void
<<Java Class>>
QWERTY
LAB3_2_polymorphism
QWERTY(int,String[],String[],String[],SpecialButtons[])

Your task is to make sure that you modified all the new specifications and the program works flawlessly.

```
1-QWERTY
2-Claculator
Please choose one of the two keyboards to run:
1
1->a  |11->k |21->u |31->4         |
2->b  |12->l |22->v |32->5         |
3->c  |13->m |23->w |33->6         |
4->d  |14->n |24->x |34->7         |
5->e  |15->o |25->y |35->8         |
6->f  |16->p |26->z |36->9         |
7->g  |17->q |27->0 |37->Space     |
8->h  |18->r |28->1 |38->New Line  |
9->i  |19->s |29->2 |39->Backspace |
10->j |20->t |30->3 |
Please enter the code of the button or -1 to exit: -1
1-QWERTY
2-Claculator
Please choose one of the two keyboards to run:
2
0->0 |10->+         |
1->1 |11->-         |
2->2 |12->/         |
3->3 |13->*         |
4->4 |14->=         |
5->5 |15->(         |
6->6 |16->)         |
7->7 |17->Space     |
8->8 |18->New Line  |
9->9 |19->Backspace |
Please enter the code of the button or -1 to exit:
```

Note: You should **NOT** change the TableViewer class, the EvaluationString class and the OperatingSystem class.

Part#3 What is the benefit from inheritance?

In the last lab, the question was (What is the difference between having inheritance in your code or not?). This lab should help you see the benefit from using inheritance in the last lab. How does inheritance improve the quality of your code? How does inheritance help improve the accessibility to your code?

What to turn in

You should turn your work in a **MS Word** file that has your codes for each part as specified below. All the codes should be yours. **NO COPY/PASTE IS ALLOWED**. You should always copy/paste the code from your compiler and take a screenshot of the console.

Part#1:

- The Keyboard class.
- The QWERTY class.
- The Calculator class.
- A Screenshot of the console after you run the main class that shows your testing.
- A txt file that has a copy of the extensive testing of simulator output

Part#2:

- The Keyboard class.
- The QWERTY class.
- The Calculator class.
- The KeyboardButton class.
- The NamedButtons class.
- The SpecialActionButton class.
- A Screenshot of the console after you run the main class that shows your testing.
- A txt file that has a copy of the extensive testing of simulator output

Part#3:

- Answer the question with examples.