Assignment 1

Summary
In this assignment you will add three methods to the Picture class (Picture.java). You will be adding methods to frame and panel pictures.

Deadline
This assignment is due on Friday, February 28 at 11:00 pm.

Description
Frame
Create a new method:
public Picture frame(int frameSize, Color frameColor)
in Picture.java. This method will create and return a new picture object. The new picture object will contain the original picture inside a frame. The thickness of the frame is given by the parameter frameSize. The color of the frame is given by the parameter frameColor. The new picture object will be larger than the original because of the frame.

Panel
Create a new method:
public Picture panel(int columns, int rows, Color dividerColor)
in Picture.java. This method will create and return a new picture object. The new picture object will be the original picture with vertical and horizontal divider lines drawn on top. The columns parameter will specify the number of column segments that will result. That is, the number of vertical lines to be drawn is one less than the number of columns. e.g. four columns require three divider lines. Likewise for rows. The dividerColor parameter will specify the color of the divider lines. Divider lines should be one pixel thick and be evenly spaced. See the examples for reference.

PanelFrame
Create a new method:
public Picture panelFrame(int columns, int rows, int frameSize, Color dividerFrameColor)
in Picture.java. This method will create and return a new picture object. The new picture object will be the original picture both paneled and framed. The columns and rows parameters determine the paneling columns and rows. The thickness of the frame is given by the parameter frameSize. The dividerFrameColor parameter will serve as both the dividerColor and frameColor.
Adding a Constructor Exercise

- Create another constructor in the Student class.
  - That takes both the name and an array of grades.
  - To create an array of doubles for the grade array use:
    ```java
double[] gradeArray = {90.3, 85.2, 89.7, 95.3};
```
- Use the debugger to check what happens during execution of this constructor.

Calculating the Grade Average

- Now that a student has an array of grades, one of the things we probably would like is to
  - Show the average when you print information about the student
- To calculate an average:
  - Sum the grades
  - Divide by the number of grades
    - The length of the array
- We need to be careful of
  - A null gradeArray
  - A 0 length gradeArray

Create a Method Exercise

- Create a method (`getAverage`) that will calculate and return the average of the grades in the grade array
  - It will return `0` if the grade array is `null`.
  - It will return `0` if the grade array length is `0`.
  - Otherwise it will return
    (the sum of the grades) / (the number of grades).
- Add to the `toString` method a call to this method.
- Use the debugger to check that this is working correctly:
  - Stop in the `getAverage` method.

Accessing Fields from Other Classes

- Fields are usually declared to be private
  - So that code in other classes can’t directly access and change the data.
Accessing Fields from Other Classes

- Try this in the interactions pane
  ```java
  Student student1 = new Student("Sue Clark");
  System.out.println(student1.name);
  ```
- You will get an exception
  - Short for exceptional event – i.e. an error
- In general outside classes cannot use
  ```java
  object.field
  ```
to access a private field value.

Accessors and Modifiers

- Rather than allowing direct access to the data fields inside a class, most classes are designed with methods to get or change the class fields’ values.

Accessors and Modifiers

- The methods to get the value of a field are called **accessors**.
- The methods to change the value of a field are called **modifiers**.

Accessors and Modifiers

- Two reasons to use accessors and modifiers is that
  - the methods can be used to control access to the information, and
  - the modifiers can check that new values have valid values
    - e.g. a test score is between 0 and 100.
    - A RGB color is between 0 and 255.
Accessors and Modifiers

- **Accessors**
  - public methods that return data.
  - They are a safe way to access the data fields in a class.
  - In such a way as to protect the data for this object, by not allowing direct access to the field.
  - Their names usually begin with “get...”
  - Syntax:
    ```java
    public fieldType fieldName()
    ```
  - Example:
    ```java
    public String getName() {
        return this.name;
    }
    ```

- **Modifiers**
  - public methods that modify the data fields in a class.
  - In such a way as to protect the data for this object.
  - Again, by not allowing direct access to the field data.
  - They usually begin with “set...”.
  - Syntax:
    ```java
    public returnType setFieldName(type name);
    ```
  - Example:
    ```java
    public void setName(String theName) {
        this.name = theName;
    }
    ```

Creating Student Accessors

- Add a method to get the name:
  ```java
  public String getName() {
      return this.name;
  }
  ```

- What about a method to get the array of grades?
  - It’s not safe to return an array in the same way that it is to return a single value.
  - If someone gets the array they can directly change the grades in the array.
  - Since an array variable is an object, and points to the actual array.
Creating Student Accessors

• It is safer to return an individual grade at an index value (e.g. the 0th grade, the 4th grade).
  – Then other classes can’t directly change the grade

```java
public double getGrade(int index)
{
    return this.gradeArray[index];
}
```

Creating Student Modifiers

• We need some public methods
  – That let other classes ask for the grade to change or the name to change.
  – Our class is responsible for making sure this happens in ways that
    • Keeps data valid.
    • Avoids errors.

Creating Student Modifiers

• Setting a grade
  – The grade must be ≥ 0
  – The gradeArray must not be null
  – The index must be < the length of the array

• Setting a name
  – You only want to allow a new name if it is null.
    • That is, no re-naming.

Name Modifier

• Setting the name only if currently null

```java
public boolean setName(String name)
{
    if(this.name == null) {
        this.name = name;
        return true;
    } else return false;
}
```
public boolean setGrade(int index, double grade) {
    if (grade < 0 ||
        this.grades == null ||
        this.grades.length <= index ||
        index < 0) {
        return false;
    } else {
        this.grades[index] = grade;
        return true;
    }
}