TOPIC 12

CREATING CLASSES
PART 2

Notes adapted from Introduction to Computing and Programming with Java: A Multimedia Approach by M. Guzdial and B. Ericson, And instructor materials prepared by B. Ericson.

Outline

- **Defining a** main method
- **Defining** accessor methods
- **Defining** modifier methods
- **Continuing the Student Class example**

Adding a Main Method

- To test new methods that we have written for a class:
  - We can use the Interactions pane in DrJava
  - We can write a separate test program (a class with a main method)
- Or, we can test new methods by adding a main method to the class we are testing, for testing purposes

Adding a Main Method

- Add a main method to the Student class, doing in it what we have been doing in the Interactions pane:

```
public static void main(String[] args) {
   Student student1 = new Student("Student 1");
   System.out.println(student1);
   Student student2 = new Student("Student 2");
   System.out.println(student2);
   …
}
```

- You can then run the main method of the Student class in DrJava

Adding a Main Method, cont.

- Add a main method to another class TestStudent, doing in it what we have been doing in the Interactions pane:

```
public static void main(String[] args) {
  Student student1 = new Student("Student 1");
  System.out.println(student1);
  Student student2 = new Student("Student 2");
  System.out.println(student2);
  …
}
```

- You can then run the main method of the TestStudent class in DrJava

Accessing Fields from Other Classes

- Fields are usually declared to be private, so that code in other classes can't directly get and/or change the data
- Try this in a new class:

```
Student student1 = new Student("Student 1");
System.out.println(student1.name);
```

  - You will get a compilation error
  - Outside classes cannot use object.field to access the field value, since it was declared to be private

Accessor Methods

- Accessor Method (aka Getter)
  - a public method that returns the value in an object's field, without changing the object in any way
  - Syntax:

```
public fieldType getFieldName()
```

  - Example:

```
public String getName()
{
   return this.name;
}
```

Modifier Methods

- Modifier Method (aka Setter or Mutator)
  - a public method that modifies an object's data
  - so that another class cannot change the field directly
  - Syntax:

```
public returnType setFieldName(type name);
```

  - Example:

```
public void setName(String theName)
{
   this.name = theName;
}
```

Modifier Methods

- Some classes do not have any modifier methods at all
  - Example: String class
- These classes are called immutable

Creating Student Accessors

- Add a method to get the name of a student:

```
public String getName() {
    return this.name;
  }
```

- Add a method to get the array of grades for a student
  - Consider this method

```
public double[ ] getGrades() {
    return this.gradeArray;
  }
```

  - It returns a reference to an ***array object***

Creating Student Accessors

- This is unsafe! Suppose a program did this:

```
double[] stuGrades;
stuGrades = student1.getGrades();
    for (int k = 0; k < stuGrades.length; k++)
        stuGrades[k] = -1.0;
```

- It is better to not have an accessor that returns a reference to an object, if you don't want to lose control over that object
- Exception: it is safe to have an accessor that returns a String, since String objects are immutable

Creating Student Accessors

- It is safer to have an accessor method that returns the grade at an index
- Why? it is of a primitive type

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

- Example of call from our example program:

```
double[] stuGrades = new double[MAX];
for (int k = 0; k < ???? ; k++)
    stuGrades[k] = student1.getGrade(k);
```

- A program would now need to know the size of the grades array. How to do that?

Creating Student Modifiers

- Our class is responsible for making sure this only happens in such a way as to keep the data valid and not cause errors
- Setting a grade at an index:
  - The grade must be >= 0
  - The gradeArray must not be null
  - The index must be valid
- Setting a name:
  - We can decide if this can be changed or not, depending on whether a name was already provided

Name Modifier

- Set the name only if currently null, and return a boolean indicating success or not:

```
public boolean setName(String theName)
{
    if (this.name == null)
    {
        this.name = theName;
        return true;
    }
    else
       return false;
 }
```

Grade Modifier

- Set the grade at an index according to our criteria:

```
public boolean setGrade(int index, double newGrade)
{
   if (newGrade < 0 || this.gradeArray == null ||
                this.gradeArray.length <= index ||  index < 0)
      {
       return false;
      }
   else
      {
       this.gradeArray[index] = newGrade;
       return true;
      }
}
```

Grade Array Modifier

- We may also want a method that sets the whole grade array if it is currently null:

```
public boolean setGradeArray(double [] theArray)
{
      if (this.gradeArray != null)
      {
       return false;
      }
    else
      {
       this.gradeArray = theArray;
       return true;
      }
}
```

Exercise: Create a Course Class

- Suppose we want to model keeping track of grades in a UWO course
- For a course we might want to know
  - The instructor's name
  - The course
  - The students in that course
    - Their names
    - Their grades in the course

Exercise: Create a Course Class

- We want fields (attributes) for:
  - Instructor's name
  - Course name
  - Students in the course
- What type should each of these be?
  - A name can be a string
  - A course name can be a string
  - The students in the course can be an array of Student objects

Exercise: Create a Course Class

- Define the Course class
- Add the attribute declarations
- Add constructors
  - A constructor that takes only the course name
  - A constructor that takes the instructor's name and course name

Exercise: Create a Course Class

- Add a toString method
- Add accessor methods
- Add modifier methods

Exercise: Create a Course Class

- **Add a new method getNumberOfStudents that returns the number of students in the course**
  - **What will be the return type?**
  - **Hint: The length of the array is not a count of the actual students**
  - **Find out how many in the array are not null**

Summary

- Main method
- Accessor methods
- Modifier methods
- **Using these in the Student Class example**