

Metropolitan State University

ICS 372 Object-Oriented Design and Implementation

Assignment 1 Classes and Relationships

Total points: 50

Objective

In this assignment, you will practice solving a problem using object-oriented programming techniques in Java; specifically, you will use the concept of **object composition**. Basically, you are asked to implement a Java application with several classes.

Problem Description

Implement three classes to keep track of a collection of triangles. You will also write a fourth class to test the implementation.

The classes are described below.

Point Class

The `Point` class represents a point on a plane. Every object of type `Point` has the following attributes.

- `x`, an integer that stores the x-coordinate of a point on a plane
- `y`, an integer that stores the y-coordinate of the same point on a plane
- `id`, an integer that uniquely identifies the `Point` object.
- A static field (say, `idCounter`) to generate the value to be stored in `id`. Increment this field once within the constructor and store the value in `id`. Since there is exactly one instance of `idCounter`, every `Point` object will have a unique value stored in `id`.

The class also has the following constructor and methods:

- A constructor that takes the `x` and `y` coordinates as parameters **in that order** and stores the two values in the respective fields. It also generates a unique `id` and stores the value in the `id` field.
- Getters for `x` and `y`.
- An override of `toString()`, which returns a properly-formatted `String` object, showing the values of `id`, `x`, and `y` coordinates, in an understandable way.
- An override of `equals()` and `hashCode()`. Two `Point` objects are equal only if they have the same `id`. All standard constraints on `equals()` must be satisfied.

Triangle Class

Every `Triangle` object has

- an array of `Point` objects of length 3. The cells store the three `Point` objects, representing the triangle vertices.
- `id`, an integer that uniquely identifies the `Triangle` object.
- A static field (say, `idCounter`) to generate the value for `id`. Increment this field once within the constructor and store the value in `id`.

Besides these fields, the class has the following constructors and methods.

- A constructor that takes three `Point` objects as parameters and stores them in the three cells of the field of the `Point` array. It also generates a unique `id` and stores the value in the `id` field.
- A constructor with no parameters and uses the first constructor to store null values in each of three cells and generate and store a unique `id`.
- A constructor with a single `Point` object as parameter and uses the first constructor to store this `Point` in the first cell (index 0) and null values in the other two cells and generate and store a unique `id`.
- A method named `setPoint()` with two parameters:
 - a) `index` – an integer that specifies the vertex to be changed.
 - b) `point` – a `Point` object to be stored in the cell corresponding to `index`.The method stores the `point` in the appropriate cell (0 1, 2) given in `index`. If `index` is anything other than 0, 1, or 2, the method does not store the `Point` object.
- An override of `toString()`, which returns a properly-formatted `String` object, showing the values of `id`, and the `Point` objects in an understandable way. Use `toString()` of `Point` appropriately.
- An override of `equals()` and `hashCode()`. Two `Triangle` objects are equal only if they have the same `id`. All standard constraints on `equals()` must be satisfied.

Triangles Class

This is a collection class that uses `java.util.LinkedList` to store `Triangle` objects.

The class has the following members.

- A field named `triangles` – a linked list of `Triangle` objects. **Use generics properly.**
- A method named `addTriangle()` with a single parameter of type `Triangle`. The return type is `void`. The method adds the `Triangle` object to the `triangles` list.
- A method named `deleteTriangle()` with a single parameter of type `Triangle`. The return type is `Triangle`. The method deletes the `Triangle` object from the `triangles` list and returns the deleted `Triangle`. If the `Triangle` object is not in the list of `Triangle` objects, the method returns `null`.

- A method named `getTriangle()` with a single parameter of type `int`. The return type is `Triangle`. The method returns the `Triangle` object with an `id` equal to the one given in the parameter. If such a `Triangle` object is not in the list of `Triangle` objects, the method returns `null`.
- An override of `toString()`, which returns a properly-formatted `String` object, showing the values of all `Triangle` objects in an understandable way. Use `toString()` of `Triangle` appropriately.

Driver class

Your driver class should do the following actions:

- Create nine `Point` objects with your choice of `x` and `y` coordinates, but every coordinate value should be unique
- Create three `Triangle` objects. Each `Point` object should be stored in exactly one `Triangle` object.
- Invoke every constructor and method of `Triangles`, `Triangle`, and `Point`, at least once, including `toString()` and `equals()`. The `toString()` methods of `Point` and `Triangle` could be indirectly invoked, via the `toString()` method of `Triangles`. Do not worry about testing `hashCode()`.

Grading

Your grade in this assignment is based on the following:

- Your submission meets specifications as described above.
- You use the same name (match the case) for all methods as specified in the above description.
- The program is robust with no runtime errors or problems.
- You follow the good programming style as shown in the D2L document [CodingStandards.pdf](#) located in the Assignments folder.
- You are encouraged to use Eclipse's menu commands to create constructors, getters, setters, `toString()`, `equals()`, and `hashCode()`. But notice that the code generated in `equals()` does not meet the specifications for naming (`obj`, instead of `object`) or use of curly brackets for code in `if` statements. It is your responsibility to take care of those issues.
- Follow the [submission instructions](#).

Specific Grading Criteria.

1. `Point`, `Triangle`, and `Triangles` classes are correctly and properly coded as specified. 24 points
2. The Driver class properly invokes all methods of the other three classes. 6 points
3. The program works correctly. 10 points
4. The program follows all coding conventions. 10 points

Submission Instructions

Follow the steps given below to upload your code to D2L:

- Create a java project and call it <yourFirstName><yourLastName>Assignment1 (e.g., mine will be called MohamedKishawyAssignment1).
- In the dialog for a module name, choose not to create a module. If you submit a project with a module and I am unable to execute it, you will lose 10% of the grade.
- Create the .java files to implement the classes as described above.
- Archive the project into **one zip** file using Eclipse using the following steps:
 - In Eclipse Project Explorer, right click on the **project folder** of the project and click on Export.
 - Choose General then Archive File and click Next.
 - Use the Browse key to choose a folder to store the archive file on your hard drive and give the file the same name as your project (e.g., MohamedKishawyAssginment1.zip), then click Save, then click Finish.
 - Upload the **.zip** file you created to the D2L folder called Assignment#1.
 - The most recent submission will be graded, unless you indicate so in the submission page.