Homework 13: Polymorphism

Release date: 4/15
Due date: 4/22 by 11:59 PM

Goals

- Learn how to declare abstract classes
  - Learn how to declare abstract methods
- Learn how to extend abstract classes
  - Learn how to override methods

Introduction

As was taught earlier in the semester, Java allows the declaration of both classes and interfaces. Being an object-oriented programming language, it also allows classes/interfaces to extend/implement others through a concept called inheritance.

An interface can be thought of as a contract. Any concrete class that implements an interface must provide declarations for methods contained in it. An abstract class is similar to an interface, in that it requires its concrete subclasses to provide declarations for its abstract methods. Abstract classes, though, can also contains fields and constructors.

The power of both interfaces and abstract classes is that types can be related, making an assertion that subtypes have certain methods in common. For example, **`AbstractList`** is an abstract class held in the **`java.util`** package. It contains a **`get()`** method that is marked as abstract. This means that its subclasses, such as **`ArrayList`**, must provide a declaration for that method.

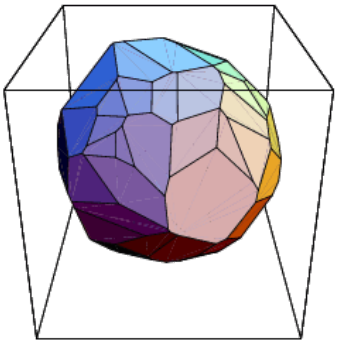
One can declare a field of type **`AbstractList`**, and will know for a fact that any reference it holds has a **`get()`** implementation. This allows an existing implementation to be swapped out for a potentially better one, and not break existing code.

You will be experiment with abstract classes through polyhedra below.

Description

Your task for this homework is to create three classes — **`Polyhedron`**, **`Tetrahedron`**, and **`Icosahedron`**. **`Polyhedron`** is an abstract class, of which both **`Tetrahedron`** and **`Icosahedron`** extend. Field, constructor, and method declarations for each are listed below.

Polyhedron



Fields

Name	Type	Access Modifier
`sideLength`	`double`	`private`

Constructor

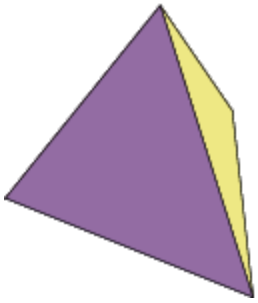
Parameters	Access Modifier
`double sideLength`[1]	`public`

Methods

Name	Return Type	Parameters	Access Modifier	Abstract
`getSideLength`	`double`	*None*	`public`	✗
`setSideLength`[2]	`void`	`double sideLength`	`public`	✗
`equals`	`boolean`	`Object anObject`	`public`	✗
`toString`	`String`	*None*	`public`	✗
`getSurfaceArea`	`double`	*None*	`public`	✓
`getVolume`	`double`	*None*	`public`	✓

[1] An **`IllegalArgumentException`** should be thrown if the **`sideLength`** argument is negative
[2] See 1

Tetrahedron



Supertype

Name
`Polyhedron`

Constructor

Parameters	Access Modifier
`double sideLength`[3]	`public`

Methods

Name	Return Type	Parameters	Access Modifier
`getSurfaceArea`[4]	`double`	*None*	`public`
`getVolume`[5]	`double`	*None*	`public`
`equals`	`boolean`	`Object anObject`	`public`
`toString`	`String`	*None*	`public`

[3] An `IllegalArgumentException` should be thrown if the `sideLength` argument is negative
[4] Surface area formula: `sqrt(3.0) * (sideLength)^2`
[5] Volume formula: `(sideLength)^3 / (6.0 * sqrt(2.0))`

Icosahedron

Supertype

Name
`Polyhedron`

Constructor

Parameters	Access Modifier
`double sideLength`[6]	`public`

Methods

Name	Return Type	Parameters	Access Modifier
`getSurfaceArea`[7]	`double`	*None*	`public`
`getVolume`[8]	`double`	*None*	`public`
`equals`	`boolean`	`Object anObject`	`public`
`toString`	`String`	*None*	`public`

[6] An `IllegalArgumentException` should be thrown if the `sideLength` argument is negative
[7] Surface area formula: `5.0 * sqrt(3.0) * (sideLength)^2`
[8] Volume formula: `((15.0 + (5.0 * sqrt(5.0))) / 12.0) * (sideLength)^3`

`equals()` and `toString()` implementations

The **`equals()`** method in each class should first determine if the specified object is an instance of the class. If it is, the side length of each should be compared for exact equality.

The **`toString()`** method in each class should return a **`String`** of the form *ClassName[sideLength]*, where *ClassName* is the class's name, and *sideLength* is the value of the **`sideLength`** field displayed with six decimal places (Hint: the format specified **`%f`** displays six decimal places by default).

JavaDoc

If you would like to view the JavaDoc for this homework, you may do so by going here. All necessary information is contained in this handout, though.

Submission

You are required to complete three classes — **`Polyhedron`**, **`Tetrahedron`**, and **`Icosahedron`** — that follow the specifications outlined above.

Submit all three **`.java`** files to Vocareum ***through Blackboard***. Keep in mind that your latest submission will be the one transferred to the gradebook.

Grading Rubric

- **`Polyhedron`** class (40 points)
  - 2.5 points
    - Making Polyhedron abstract
    - Declaring the **`sideLength`** field
  - 5 points for each constructor/method declaration (7 total)
- 30 points
  - **`Tetrahedron`** class
    - 3 points for extending **`Polyhedron`**
    - 5.4 points for each constructor/method declaration (5 total)
  - **`Icosahedron`** class
    - 3 points for extending **`Polyhedron`**
    - 5.4 points for each constructor/method declaration (5 total)