OOP in Java

Jared F Bennatt

February 6, 2020

I implemented a **RealFunction** interface which represents a real value function that returns a scalar (maps from $\mathbb{R} \mapsto \mathbb{R}$). This interface also includes two overloaded methods for integrating a function (*abstraction* and *polymorphism*). There is a **Polynomial** class which directly implements **RealFunction** (*inheritance*) and a **Quadratic** class which extends Polynomial (*inheritance*). Inside of the **Polynomial** class, the polynomial itself is stored as an array of coefficients which is hidden from the user (*encapsoluation*). Finally there is a **QuadraticException** which handles cases inside the **Quadratic** class where 1) there are no real solutions, 2) there are infinite real solutions, and 3) infinite solutions give no y-intercept.

- Inheritance

  Polynomial inherits functionality from the **RealFunction** class and **Quadratic** inherits from the **Polynomial** class (and inherits its evaluation implementation).

- Abstraction

  All of the methods are cases of abstraction. Particularly the **integrate** in **RealFunction**, **eval** in **Polynomial**, **integrate** in Polynomial, and all of the methods in **Quadratic** which give the two solutions and the y-intercept.

- Encapsulation

  The two cases of encapsulation are in the **Polynomial** class where I store the values of the coefficients in an internal array, invisible to the user and in the **Quadratic** where I store the values used to implement the methods that give the solutions and y-intercept.

- Polymorphism

  All of the duplicate constructors are examples of overloading-polymorphism. The **eval** function is an example of overriding-polymorphism. I don't think the **integrate** in the **Polynomial** class is an example because, although it has the same name as the *static* method in the **RealFunction** class, it's an instance member of **Polynomial** (and doesn't override the static **integrate** from **RealFunction**)..