Liz Brondos

Foxes and Rabbits

Rabbit: Hello. I am a little innocent hopping rabbit. For some reason, silly textbooks used for college classes like differential equations and computer science like to use us in predator-prey problems. We get to be the prey.
Fox: The same textbooks like to say that we foxes are predators of the rabbits. But you know what? We don't even eat rabbits! Yuck, they taste like mud! Mice are much more appetizing. The writers of those books really need to do some research.
Rabbit: This particular computer science textbook describes a foxes-and-rabbits simulation that is supposed to imitate real life and keep track of rabbit and fox populations. But come on, how realistic can you be? You can't just predict population growth and decay just like that. I mean, what if there was a meteorite that crashed down on our forest? Or what if the giant Easter Bunny came by and killed off all the foxes? Or what if—
Fox: Enough of that! Let's get back to the simulation program. The program consists of several different classes, and three of them are Fox, Rabbit, and Simulation. Rabbit keeps track of the breeding information of rabbits, Fox keeps track of information about fox hunger and how fast we eat the rabbits. Simulation shows how we interact in a field.
Rabbit: Oh be quiet, Fox, you are such a nerd.
Fox: No, I will not be quiet! The simulation program is all right as it is, but we could really make it have a better class structure. We are going to make an abstract superclass called Animal that will include both foxes and rabbits and define common attributes such as age, alive, and location.
Rabbit: Abstract?
Fox: Yes, an abstract class is not intended for creating instances, but instead its purpose is to serve as a superclass for other classes. Abstract classes can contain abstract methods.
Rabbit: What on earth are you talking about?
Fox: For a subclass of an abstract class to become concrete, it has to provide implementations for all inherited abstract methods; otherwise it will be abstract itself.
Rabbit: Oh, whatever. I hate it when you start talking like a smarty-pants. Sometimes I wish a hunter would just come shoot you.
Fox: Oh yeah, that reminds me! We want to add a hunter to the program too. So we create an Actor class that will include Animal and Hunter as its subclasses. Humans, foxes, and rabbits are all Actors.
Rabbit: You mean I get to be a movie star?
Fox: No, you're too stupid. Anyway, we also create a class called Drawable to be a superclass of all the drawable actors. Some subclasses of Actor are also subclasses of Drawable. When one class inherits from more than one superclass, we call this multiple inheritance.
Rabbit: You know what you inherited from your parents? A big fat mouth and big ego because you think you're so smart. I want to throw a carrot into your face!
Fox: Oh, that reminds me of an interface! A java interface is a specification of a type that doesn't define any implementation for the methods.

Rabbit: You're just trying to show off! You think you're better than everyone else, don't you?
Fox: You know, you are being such a pain that right now I wouldn't mind eating you. I don't care if you taste like mud.
(The fox opens up his mouth to eat the rabbit, but at that moment a meteorite falls down on the forest and the simulation program crashes.)