



Computer Science Virtual Learning

HS Computer Science A

May 18th, 2020

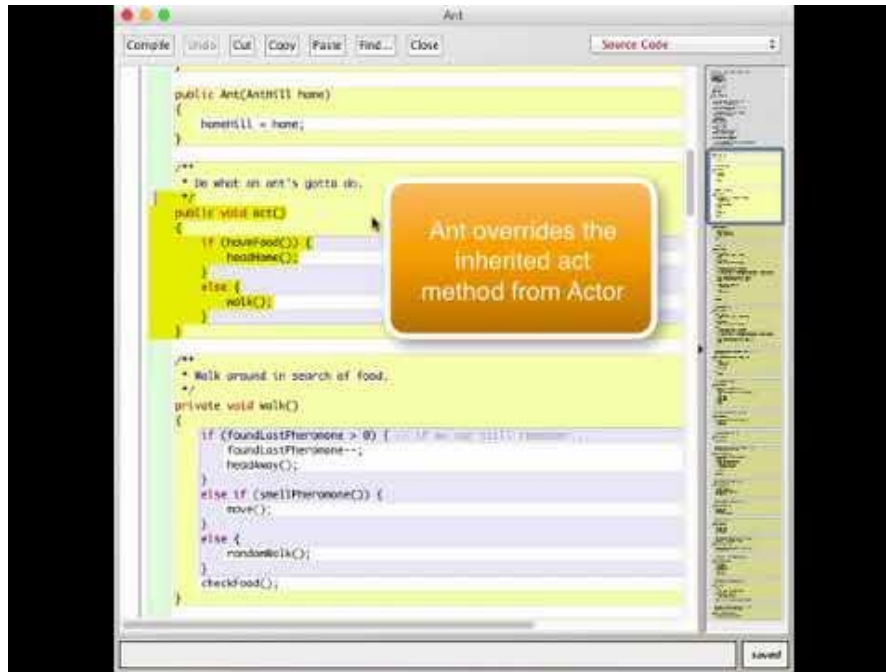


Lesson: **What is Object-Oriented Programming?**

Objective/Learning Target:

Understanding what Object-Oriented Programming is and how to apply it using Java

What is an Object-Oriented Programming?



```
public Ant(AntHill home)
{
    homeHill = home;
}

/**
 * Do what an ant's gotta do.
 */
public void act()
{
    if (haveFood()) {
        headHome();
    }
    else {
        noWork();
    }
}

/**
 * Walk around in search of food.
 */
private void walk()
{
    if (foundLastPheromone > 0) {
        foundLastPheromone--;
        headWay();
    }
    else if (shellPheromone() {
        move();
    }
    else {
        randomWalk();
    }
    checkFood();
}
```

Ant overrides the inherited act method from Actor



What is Object-Oriented Programming?

Object-oriented programming has three main features: objects, inheritance, and polymorphism.

Objects have data (fields) and behavior (methods) and do the work in an object-oriented program. Objects are created by classes. A class defines the data (fields) and behavior (methods) for all objects of that class. You can create many objects from the same class.



What is Inheritance?

Inheritance allows for cleaner code since a class can inherit fields and behavior from another class instead of copying code from class to class. The parent class is specified using the `extends` keyword in the class declaration. The class that is extending the parent class is called the child class. In the ants scenario the `Ant` class inherits from the `Actor` class. The `Ant` class is the child class and the `Actor` class is the parent class. The `Ant` class inherits the `act` method from the `Actor` class, but overrides it by creating a method with the same signature that will be executed instead of the parent's method. This allows the `Ant` class to modify what an `Ant` object does when it acts.

What is Polymorphism?

Polymorphism allows for specialized behavior based on the run-time type. It also removes the need for conditional execution based on the type. Java uses inheritance-based polymorphism where a parent class has a method that the children classes override to provide specialized behavior. In the Ant scenario the Balloon and Bomb classes inherit from the Actor class and both override the act method. The world contains a list of all Actor objects in the world and tells each to act. What happens when an Actor object acts depends on the class that created it (the run-time type).





Objects and Classes

In object-oriented programming the programmer writes a class which defines what all objects of the class know and can do. You can think of the class as like a cookie cutter or factory that produces objects. All objects created by the same class have the same fields and methods. A field is something the object knows about itself and a method is a thing the object can do.

A class also has constructors which initialize the fields when the object is created. A class can also have a main method which can be used to test the class.



Check Your Understanding: Person Class

What should we want to know about a person? What we want to know depends on what problem we are trying to solve. In one situation we might want to know the person's name and phone number and email. We need ways to get and set the data (fields) so we create getters and setters.

Go to:

<https://runestone.academy/runestone/books/published/apcsareview/OOBasics/objectsAndClasses.html>

Scroll to the Person Class section and Modify the code above to add more constructors. Also modify the main method to test the new constructors.

A screenshot of a code editor interface. At the top right, there is a "Run" button and a progress indicator showing "Original - 1 of 1". The code is as follows:

```
1 public class Person
2 {
3     // fields
4     private String name;
5     private String email;
6     private String phoneNumber;
7
8     // constructor
9     public Person(String theName)
10    {
11        this.name = theName;
12    }
13
14    // methods - getters
15    public String getName() { return this.name; }
```

The output area below the code shows:

```
Sana null null
Jean jean@gmail.com 404 899-9955
```

At the bottom of the editor, it says "Activity: 1 -- ActiveCode (PersonObjExample)".

Check Your Understanding: Die Class

What if you wanted to represent a die which has 6 sides and you can roll it? You might also want to keep track of the last value rolled. Does the following class have everything it needs?

Go to:

<https://runestone.academy/runestone/books/published/apcsareview/OOBasics/objectsAndClasses.html>

Scroll to the Die Class section. Can you modify the Die class to keep a record of all the values this dice has rolled? How would you do that?

The screenshot shows a code editor window with a yellow background. At the top right, there is a "Run" button and a progress indicator labeled "Original - 1 of 1". The code is as follows:

```
7     lastValue = (int) (Math.random() * 6) + 1;
8     return lastValue;
9 }
10
11 public static void main(String[] args)
12 {
13     Die d = new Die();
14     for (int i = 0; i < 10; i++)
15     {
16         System.out.println(d.roll());
17     }
18 }
19 }
20
```

Below the code editor is a scrollable output area with a light gray background. It contains the following text:

```
3
6
1
6
6
1
1
5
3
6
```

At the bottom of the editor, it says "Activity: 2 -- ActiveCode (DieExample)".



Check Your Understanding: Coin Class

What if you just wanted to simulate flipping a coin? What would you need the objects of the class to know and do? You would want to flip the coin (set it randomly to heads or tails) and ask if the value is heads or tails. See the class below for one way to do this. Notice the use of a constant for HEADS. Any field that is declared to be static can't be changed and so is a constant.

Go to:

<https://runestone.academy/runestone/books/published/apcsareview/OOBasics/objectsAndClasses.html>

Scroll to the Coin Class section. Modify the class to add a `isTails` method that returns true when the value is not heads.

A screenshot of a Java IDE. At the top right, there is a green "Run" button and the text "Original - 1 of 1". The code editor shows the following Java code:

```
1 public class Coin
2 {
3
4     // constant to represent heads
5     private static int HEADS = 1;
6
7     // current value of the coin
8     private int value = 0;
9
10    // method to randomly set the value of the coin to heads or tails
11    public void flip()
12    {
13        if (Math.random() < 0.5)
14        {
15            value = 0;
```

The output window below the code shows a sequence of 15 results: Tails, false, Tails, false, Tails, false, Heads, true, Heads, true, Tails, false, Tails, false, Heads, true, Tails, false, Heads, true. On the right side of the output window, there are two buttons: "Annotate" and "Highlight".



For More Resources and to Check Answers

Go to: <https://runestone.academy/runestone/books/published/apcsareview/OOBasics/objectsAndClasses.html>