**1.2.4 Cool Cat – For Loop Practice**

**Introduction**

In this assignment we will modify the Cool Cat Start program that animates a piano-playing cat. While we modify it, we will get more proactive with for loops and will abstract our loops into some new methods to make our program DRY!

Concepts covered are:

* More abstraction
* More For loops
* DRY (Don't Repeat Yourself)
* Keywords: **for**

**Materials**

* Computer with Greenfoot

**DRY with For Loops**

1. Download cool-cat-start.zip and Save a Copy As cool-cat-with-loops in your project folder. Open your cool-cat-with-loops scenario. This is the scenario you will develop following instructions below.
2. On the cat object placed in CatWorld, call the playPiano() method.
   * Watch the cat play the piano.
   * Check out the play method’s source code.
   * Describe what is happening in the body of the play method.
   * What method calls are being made?
   * Check for patterns. What patterns do you see?
3. In this exercise, we are going to try to make the code more **DRY (Don’t Repeat Yourself)** so we need to find the three lines of code that repeat the most. We will turn these repetitive lines into **for** loops. We can make the cat do the same thing with a **for** loop in which we use each of the calls identified above.

|  |
| --- |
| * Comment out lines 71 to 102 * Start a **for** loop on the line ABOVE the second call to play sound on line 104.   Remember, the syntax is:  **for** (int n = 0; n < 8; n++)  {  code;  code;  code;  }  Inside the curly braces write the three lines that repeat so much.   * Test your code. |

1. Is there another opportunity to replace repetitive lines of code with a **for** loop?

|  |
| --- |
| * Replace other repetitive lines with another **for** loop. * Test your code. |

1. You’ll notice that the two loops we wrote are very similar except in the number of times they repeat. Instead of writing two almost identical for loops, we can **abstract** the loops by writing a method called twoPlayMoves(). The method’s signature will have return type **void** and one parameter, let’s call it howMany of type **int**.

|  |
| --- |
| * Add the twoPlayMoves() method to class Cat * The abstracted method will let you enter the number of times it should loop as the parameter howMany when you call it. * Here’s the code:   **for** (int n = 0; n < howMany; n++)  {  setImage("cat-play-3-piano.png");  wait(6);  setImage("cat-play-4B-piano.png");  wait(6);  }   * Replace the two for loops you wrote with calls of the twoPlayMoves() method. Remember to call the proper argument for howMany each time. |

1. Let’s improve the program some more:

|  |
| --- |
| * Write a for-loop to replace the two occurrences of spinning to play (and spinning away from playing). * Write a method whose code is the for loop that manipulates the files "spin-1.png", "spin-2.png", "spin-3.png", "spin-3-piano.png", "spin-4-piano.png", "spin-5-piano.png". * You might want to rename the images to create a more streamlined pattern. * Name your method spinAwayToPlay(). The method does not return a value and takes a **boolean** parameter which indicates whether the cat spins to start playing or not. |