AP Computer Science with Data Structures

Lab: Java Circus
Part 2

I Have No IDE

You have now succeeded in editing a Java file with a text editor and compiling it from the command line. This process can be somewhat tedious (especially using NotePad!). From here on out, you are strongly encouraged to use a Java editor or any Java IDE (Integrated Development Environment). Here are a few free products you might try:

- **TextPad (not free) is** primarily an editor, but you can also use it to compile your programs. (You can hit control-1 to compile your code and control-2 to run it, if you prefer not to use the command line.) TextPad is that rare tool that does what you want and nothing more, and hence doesn't get in your way.

- **BlueJ** is a simple and friendly cross-platform IDE aimed at beginners (and written in Java). It is very popular in high school computer science classrooms. Unlike any other Java tool I know, it lets you create objects interactively, look inside them, call methods on them, etc.

- **JCreator** is a free IDE resembling professional Microsoft or Borland products. But you may find yourself tripping over its workspaces and projects. If you do like this sort of tool, you may be interested to know that real professional developers swear by (and at) two free IDEs called **NetBeans** and **Eclipse**, which are much better than JCreator, but also much more serious and scary.

Go install and try out at least one of these development environments.

Exercise: Applause

Notice that your `AcrobatTest` class uses `System.out.println` lines to print information to the screen. Modify your `clap` method so that it *recursively* prints out "<*name*> claps" (where <*name*> is the name of your `Acrobat`) a total of `n` times, along with incrementing `count` appropriately. Test that your code runs correctly.

Exercise: Feeling Loopy

Java supports many fancy constructs, such as the "for loop". The following code will print out the word "Hello" 10 times.

```
for (int i = 0; i < 10; i++)
{
  System.out.println("Hello");
}
```

Use this idea to modify `kneeBend` to print "<*name*> knee-bends" a total of `n` times, along with incrementing `count` appropriately.

A Free Data Structure

Java has a library of built-in classes. One of the most useful of these is `ArrayList`, a data structure that lets you store an arbitrarily long list of values. The following code demonstrates some of the things an `ArrayList` can do. Study it carefully!

```
ArrayList<String> stuff;
stuff = new ArrayList<String>();
stuff.add("life");
stuff.add("the universe");
stuff.add("everything");
int howMany;
howMany = stuff.size();      //howMany is now 3
String home;
home = stuff.get(1);         //home is now "the universe"
stuff.remove(1);             //"the universe" has been removed
```

Here's a summary of some of the messages you can pass to an `ArrayList`.

```
class java.util.ArrayList<E>
    • int size()
    • boolean add(E x)
    • E get(int index)
    • E set(int index, E x)
           // replaces the element at index with x
           // returns the element formerly at the specified position
    • void add(int index, E x)
           // inserts x at position index, sliding elements
           // at position index and higher to the right
           // (adds 1 to their indices) and adjusts size
    • E remove(int index)
           // removes element from position index, sliding
           // elements at position index + 1 and higher to the
           // left (subtracts 1 from their indices) and adjusts size
           // returns the element formerly at the specified position
```

When you use an `ArrayList`, you must specify the type of objects that will be contained within the `ArrayList`. For example, we can create an `ArrayList` containing only `Rabbit`s, as follows:

```
ArrayList<Rabbit> bunnies;
     //declares a variable bunnies to be capable
     //of pointing to an ArrayList of Rabbits

bunnies = new ArrayList<Rabbit>();
     //constructs a new, empty ArrayList capable of containing
     //Rabbits, and associates the variable bunnies with it.
```

The Java class library is arranged in *packages*. Built-in classes such as `Object`, `String`, and `System` appear in the `java.lang` package. When you use these classes in your code, Java already knows to look inside this package to find them. On the other hand, notice that the `ArrayList` class (and many others we'll be using) appears in the `java.util` package. If you just start referring to `ArrayList` in your code, your compiler will give you one of these errors.

```
cannot resolve symbol
symbol  : class ArrayList
location: class MyClass
```

One way to fix this problem is to modify your code to refer to `ArrayList` by its full name each time it appears. Here's what that would look like.

```
java.util.ArrayList<Rabbit> bunnies;
bunnies = new java.util.ArrayList<Rabbit>();
```

As you can see, this is tedious and ugly. A better solution is to *import* the `ArrayList` class at the top of each file that uses it. Here's how that would look.

```
import java.util.ArrayList;

public class MyClass
{
    ... bunch of code which can directly refer to ArrayList ...
}
```

Alternatively, you can import the *entire* `java.util` package in one fell swoop as follows.

```
import java.util.*;
```

It's really just a matter of taste as to which import style you prefer.

Exercise: Joining the Circus

Implement a class called `Circus`. Its `join` method should add an `Acrobat` to the circus. Its `clap` method should prompt all acrobats in the circus to clap a specified number of times. Its `kneeBend` method should behave similarly. Finally, its `count` method should report the total number of exercises performed by all acrobats in the circus. The following example shows the `Circus` class in action.

```
Circus circus;
circus = new Circus();
circus.join(new Acrobat("Alice"));
circus.clap(2);        //Prints "Alice claps" twice.
circus.join(new Acrobat("Bobbo"));
circus.kneeBend(3);    //Prints "Alice knee-bends" three times
                       //and "Bobbo knee-bends" three times.
System.out.println(circus.count());   //Prints 8.
```

Be certain that your code is properly documented.

Show your teacher your circus act!