Using Objects and Problem Solving

Overview

Using Objects
- Data types
- Declaring variables
- Creating objects
- Calling methods
- ArrayList
- String
- Memory

Problem Solving

Using Objects

Data types

A *data type* is a set of values and a set of operations that can be performed on those values.

There are eight *primitive types*:

byte, short, **int**, long
float, **double**
boolean
char

There are an infinite number of *object types*:

Array types:

int[], boolean[], double[][], …

Classes built by others:

String, Color, …

Classes you define yourself:

Die, Beetle, …

Declaring variables

Object types are just like other types:

```
Sword sting;
Dragon smaug;
Ring[] dwarfRings;
```

Creating objects

Use `new`:

```
smaug = new Dragon();
```

This calls a constructor. Some constructors require arguments:

```
tan = new Color(210, 180, 140);
```

There are just a few special cases for creating an object without using `new`:

```
int[] numbers = {4, 8, 15, 16, 23, 42};

String magicWords = "Klaatu barada nikto";

String lunch = "hot" + "dog";
```

Calling methods

In the procedural paradigm (C, FORTRAN, COBOL), the world is made of data structures and you *do things to them*.

In the object-oriented paradigm (Smalltalk, C++, Java), the world is made of objects and you *tell them to do things.*

To call a method on an object:

object expression`.`*method name*`(`*arguments*`)`

For example:

```
enterprise.fireTorpedoAt(klingons[i]);
new BeetleGame().run();
```

You call an instance method on an instance, but a static method on a class.

ArrayList

This handy class, in the java.util package, acts like a stretchable array. It has many useful methods.

```
ArrayList<String> groceries = new ArrayList<String>();
```

(This one is an ArrayList of Strings. More on that in the lesson on generic types.)

```
groceries.add("eggs");
groceries.add("bread");
StdOut.println(groceries.size()); // Prints 2
StdOut.println(groceries.toString()); // Prints [eggs, bread]
StdOut.println(groceries.contains("eggs")); // Prints true
groceries.remove("eggs");
```

See the API for much more!

String

String is a class!

Strings are immutable (cannot be changed), so almost all of the methods return values (often new Strings). If `s` is a String:

```
s.length()
s.toUpperCase()
s.trim()
s.indexOf('e')
s.endsWith("ism")
```

Puzzle: the String method `getChars` has a return type of void. Since Strings are immutable, it can’t modify the String on which it’s called. What’s the point?

Memory

Memory leak

Dangling pointer

Garbage collection

Problem Solving

Don’t give up!

Take risks and make mistakes

Identify goals and obstacles

Apply prior knowledge (or find the knowledge you need)

Does context matter?

Use the resources at your disposal

Review

Classes are data types, just like the built-in ones.

Create object with `new`, then call methods on them.

There are many useful built-in classes, including ArrayList and String.

Java lets you avoid a lot of nasty memory problems.

Problem solvers press on and find the tools to get the job done.