More Complex Data Representations

Arrays

- What if you have a collection of objects of the same type, e.g. stones of different weights. How can you work with this data efficiently?
- Current approach:

```
int stoneWeight1 = 3;
int stoneWeight2 = 54;
int stoneWeight3 = 7;
…
```

Cumbersome!

Hard to access particular entries!

Arrays

- Arrays: indexed list of items
  - All items must have the same type
  - Identified by a name and index
  - ***Index starts at 0***
- e.g. int [ ] stoneWeight = {3, 54, 7};
  - The value of stoneWeight[0] is 3
  - The value of stoneWeight[1] is 54, etc.
- Can use like any other variable. e.g. in assignment:
  - stoneWeight[2] = 6 + stoneWeight[1];

Arrays

Optional initialization

- int [ ] stoneWeight = {3, 54, 7};
- General form of declaration

<type> [ ] <name> = new <type>[<size>];

- e.g.
  - int [ ] intList = new int [42];
- Arrays have a fixed size

Color Array Example

Array Length

- Can use .length on any array to find how many items it can hold

```
int [] a = new int[10];
println(a.length);
```

- Note: length is not a method, so no ( )
  - Not .length( );
- What is the value of a.length?
- What is the index of the last element in the array?
- a.length is 10, and the last element is a.length-1 or 9

Manipulating Arrays

- To grow array
  - Could allocate a larger array and copy first to it
- Processing functions to manipulate arrays:
  - shorten();
  - concat();
  - subset();
  - append();
  - splice();
  - expand();

More Processing Methods for Arrays

- sort();
- reverse();

Array Functions

- Functions do not modify the original array
  - They return a new array
- For example:

```
String[] sa1 = { "OH ", "NY ", "CA "};
String[] sa2 = shorten(sa1);
println(sa1); // 'sa1' still contains OH, NY, CA
println(sa2); // 'sa2' now contains OH, NY
```

- Can update the array by assigning the result to it. e.g.

```
sa1 = shorten(sa1);
```

Rect Array Example

2D Arrays

```
int [][] intArray =    {  {1, 2, 4},
                          {5, 1, 7},
                          {2, 9, 18} };
```

The value of intArray[1][2] is 7.

Objects

- Contain *data* AND *methods*
  - Encapsulate a particular concept
    - e.g. Star
    - Both data and operations related to that entity
- Way of thinking about problems and organizing solutions
  - Object-Oriented programming

Objects

- Objects are data types
  - User definable
  - Can use like a variable name

Object vs. Class

- *Class* is the definition
- *Object* is a specific instance of that definition
- Like the difference between integer, the general type, and a specific variable of type integer

Example: Car

- We know some things about a car:
  - Mileage
  - Color
  - Model
  - Weight

- And a car can do some things:
  - Accelerate
  - Brake
  - Steer

Create a Car Class

```
class Car
{
   //data about car
   float mileage;
   color color;
   String model;
   float weight;
   //functions car can perform
   void Accelerate();
   void Brake();
   void TurnLeft();
   void TurnRight();
   float GetWeight();
};
```

Work With the Class

- create an instance of the class. i.e. an object

```
Car myBeatle = new Car();
```

- Call functions of the object

```
myBeatle.Accelerate();
myBeatle.TurnRight();
```

- Use data in the object

```
println("My car weighs " + myBeatle.weight);
```

Example

```
class foo
{
  int val = 0;
  foo(int v)
  {
    val = v;
  }
}
foo testObj = new foo(5);
```

- "foo" is a class
- "testObj" is an object
  - An instance of the class "foo"

Use . To Access Object Members

- Same for methods or variables. e.g.

```
foo testObj = new foo(5);
testObj.val = 15;
//The . accesses the val member of the foo class
```

Objects

- Object initiation
  - Constructor
- Naming convention
  - Prefix all member variables with m_

Objects Support Encapsulation

- Hide details that the world doesn't need to know about.
  - e.g. car driver doesn't need to know how the carburetor works, just how to use the brake, accelerator and steering wheel
- Simplifies what the user needs to know
- Allows details to be changed later as nothing outside depends on them

Objects are Passed by Reference

- See code example

For This Course

- Not expected to write your own classes
  - Encouraged to try
- Will use classes defined by other people
  - e.g. when working with images and video

Star Object

Can have arrays of objects…