Classes and Objects

"Class"

- Java uses the word "class" to mean two different things:
  - collection of methods (like a code library)
  - description of an object data type (a real "class")
- So far, we have created only the code-library variety.
  - All methods were just functions that were put in a class for organization.

Classes and Objects

- A "class" is a description of an object type.
  - A class is "instantiated" to create an object.
- An object is a "variable" that can contain:
  - methods (functions)
  - data (variables)
- eg. we could create a class to hold student records.
  - possible methods: `getGrade(course)`, `setGrade(course)`
  - contained data: list of grades, name, student number, …
  - create instances for each student in the class

Object Example

- We have already seen the `String` class:

```
// constructor:
String s = new String("example");
// String methods:
char c = s.charAt(3);
    // c=='m'
String s1 = s.concat("123");
    // s=="example" && s1=="example123"
boolean b = s.startsWith("ex");
    // b==true
```

Creating Instances

- A class is a description of a kind of object.
- To create an actual object than can hold data and contain methods, we must "instantiate" the class.
  - Instantiating a class creates an object.
- Each class has one or more "constructors".
  - each takes different arguments
  - lets you create an object with initialization values

Example Constructors

```
String s1;  // create String reference
s1 = new String();
    // instantiate, holds empty string
    // s1 refers to new instance
String s2 = new String("abc");
    // initialize to a literal string
char[] chars = {'a', 'b', 'c'};
String s3 = new String(chars);
    // initialize from array of characters
```

References and Instances

```
String course;
```

course: ⊡→•

```
new String("CMPT 125")
```

CMPT 125

```
course = new String("CMPT 125")
```

course: ⊡→ CMPT 125

Methods

- Methods are functions contained in a object.
- An object's contents denoted with the "`.`"
  - eg.
    ```
    String s = new String("abc");
    s.startsWith("ab") == true
    s.charAt(1) == 'b'
    ```
  - both are methods: `String` has no public variables.
- An object's methods operate on its current value
  - might change it; might just use it like an argument

Class Variables

- Variables can also be contained in an object.
  - sometimes called "properties" or "data members"
- used to keep track of "state" of the object
- Typically not visible from outside: declared as `private`.
  - Possible but not common:
    ```
    MyObj o = new MyObj();
    o.dataMember = 6;
    ```

Getters and Setters

- The state of an object is usually manipulated with methods.
  - Instead of directly accessing data members.
  - Lets the object keep its state consistant.
    ```
    Student s = new Student("30000-1234");
    s.setFirstName("Rudiger");
    System.out.print( s.getFirstName() );
    ```
- Methods that read and write data members are called "accessors" and "mutators".
  - … or "getters" and "setters".

"Static"

- Two types of methods in a class: code library functions; methods in an object description
  - How do we tell which is which?
- The `static` keyword is used to mark them.
- The short version:
  - `static`: library function
  - non-`static`: object method

Static Methods

- Won't be copied with each new object created.
- Can be called as functions, no objects needed:
  ```
  x = methodName();
  x = ClassName.methodName();
  ```
- Need all values passed as parameters.
- Used to create code-library-type classes.

Non-Static Methods

- Will be copied into each new object instance.
- Must be called as methods of an object instance:

```
MyClass o = new MyClass();
x = o.methodName();
```

- Implicitly use the object they're a member of as an argument.
- Used in classes that describe an object type.

Hello World, again