CS1007 lecture #10 notes

tue 26 feb 2002

- news
- objects and classes: formal definitions
- methods: formal definitions and details
- object relationships
- method overloading
- reading: ch 4.1-4.6

news.

- grades are posted in courseworks
- CS open house

  If you are considering the CS major, please come to the
  Computer Science Department Open House
  Wed Feb 27, 4:30-5:30pm
  In the CLIC Lab (486 CS, enter through Fairchild)
  Followed by pizza in the CS lounge!

objects.

- objects have:
  - state
  - set of behaviors
- example: a robot
  - state
    - where it is
    - where it was a minute ago
    - how fast its motors are turning now
    - how fast its motors can turn
  - behaviors
    - turn
    - go forward
    - go backward
    - stop

classes (1).

- define objects
- are “blueprints” for creating *instances* of objects
- example: a house
  - class = architect’s blueprint
  - instance = a house built following that blueprint
- *instantiate* = to build the house
- you can build MANY houses using the same blueprint, so you can instantiate many objects using the same class

classes (2).

- contain *members*:
  - data declarations
    - constants
    - variables
  - methods
    - *constructor* is special method used to *instantiate* an object of that class
    - some methods may change the values of the variables
    - some methods may *return* the values of the variables
  - scope
    - *local* vs *global*
    - *instance data*
    - *method data*

encapsulation and visibility.

- objects should be self-contained and *self-governing*
- only methods that are part of an object should be able to change that object's data
- some data elements should not even be seen (or visible) outside the object
- *public* data elements can be seen (i.e., read) and modified (i.e., written) from outside the object
- *private* data elements can be seen (i.e., read) and modified (i.e., written) ONLY from inside the object
- typically, **variables** are **private** and **methods** that provide access to them (both read and write) are **public**
- typically, **constants** are **public**
- example: house
  - walls provide privacy for the inside
  - windows provide public viewing of some of the inside

method declaration.

- like a variable, has:
  - data type:
    - primitive data type, or
    - class
  - name (i.e., identifier)
- also has:
  - arguments (optional)
    - also called *parameters*
    - *formal parameters* are in the blueprint, i.e., the method declaration
    - *actual parameters* are in the object, i.e., the run time instance of the class
  - throws clause (optional)
    (we'll defer discussion of this until later in the term)
  - body
  - return value (optional)

method use.

- program control jumps inside the body of the method when the method is *called* (or *invoked*)
- arguments are treated like local variables and are initialized to the values of the calling arguments
- method body (i.e., statements) are executed
- method *returns* to calling location
- if method is not of type *void*, then it also *returns* a value
  - return type must be the same as the method's type
  - calling sequence (typically) sets method's return value to a (local) variable; or uses the method's return value in some way (e.g., a print statement)

object relationships.

- are hierarchical
- example:

```
java.lang.Object
   |
   +--java.lang.Number
          |
          +--java.lang.Integer
```

- *is-a* relationship
  - an object that is an instance of a class
  - an `Integer` is a `Number`, which is a `Object`
  - children *inherit* properties of their parents; formally called *inheritance*
- *has-a* relationship
  - if an object declares data whose type is also a class

method overloading.

- using the same method name with formal parameters of different types
- example:
  - `java.lang.System` has a variable called `out`
  - which is a `java.io.PrintStream`
  - whose declarations include:

```
public void println();
public void println( boolean x );
public void println( char x );
public void println( char[] x );
public void println( double x );
public void println( float x );
public void println( int x );
public void println( long x );
public void println( Object x );
public void println( String x );
```