Introduction to Java
Lecture 2 (7/4/2006)

Useless Fact of the Day

• A googol is $10^{100}$ - in other words, a one followed by 100 zeros. This number is larger than the number of atoms in the entire universe.

• A googolplex is $10^{gogol}$ - a one followed by a googol zeros. It is not possible to write down a googolplex in decimal notation.

Topics

• Object-oriented-ness
• Terminology
• Anatomy of a Class

Object-Oriented Programming

• We divide a program up into “objects”
• Objects can store information, modify/calculate things, or both
• Objects interact with each other
Object-Oriented Programming

- Example: **objects** in this room
- **presentation**: stores information; allows a **person** to read the information
- **notebook**: stores information; also allows a **person** to write information into it and read info from it
- **person**: stores information; can write/read the info; can also perform calculations and modify the information it stores

Object-Oriented Programming

- Another example: **objects** in a forest
  - **tiger**:
    - stores data - color, weight, hungriness
    - does things - catch(a rabbit), eat(a rabbit)
  - **rabbit**:
    - stores data - color, weight, hungriness, trickiness, tastiness
    - does things - eat(a grass)
  - **grass** (a patch of)
    - stores data - amount of grass

Terminology

- A type of object is defined by a **class**
- An actual object is an **instance** of the class
- **Data members** are pieces of information stored by a class
- A **method** (or **function**) is a set of instructions for “doing stuff” with data
- An object has data members, methods, or (usually) both

Terminology - Data

- Any piece of data always has a **type**
  - Examples:
    - Integers - **int** - 1, 2, 5, 3423, 423094
    - Doubles - **double** - 1.32, 5.5, 2234.3634
    - Boolean - **boolean** - true, false
  - A **variable** is a piece of data with a name
    - Examples:
      - “int x;” creates an integer variable named x
Terminology - Data

- Example: a forest
  ```java
  public class Grass {
    public int amount;
  }

  public class Rabbit {
    public int weight;
    public int hungriness;
  }
  ```

Terminology - Methods

- Methods have parentheses
  - We could run a method named “go” with the line “go();” (if “go” needs no arguments)
  - **Arguments** are pieces of data given (passed) to a method; they go in the parentheses
  - The code “blah(x);” runs the method named “blah,” passing it the variable named “x”
  - The **return value** of a method is the result of its actions

Terminology - Methods

- When we define a method, we have to say what type of data will be passed in, and what type of data will be returned
- A placeholder for data passed in is called a **parameter** (an actual piece of data we pass in for a parameter when we run the method is an argument)
- Example:
  ```java
  int add(int x, int y) {
    return x + y;
  }
  ```

Terminology - Classes

- When we define a class, we give it a name, and say what data and methods it has
- We use the name of the class to create one or more **objects** whose type is that class
- Example: “Rectangle” is a class defining the attributes of rectangle objects. We can make a rectangle object called “r” with the code “Rectangle r(0, 0, 20, 20);”
- You can access the data in an object with the **dot** operator
  - Example: You could access the rectangle r’s width with the code “r.width”
Terminology - Classes

- Every class has a **constructor**
- Class X's constructor is a special method that is run whenever a new object (instance) is created whose type is class X
- Example: When we create a rectangle with "Rectangle r(0, 0, 20, 20);", we run the Rectangle constructor method, giving it the arguments 0, 0, 20, 20

Example: a forest
```java
public class Grass
{
    public int amount;
    public Grass() { amount = 50; }
}
public class Rabbit
{
    public int weight;
    public int hungriness;
    public Rabbit(int w, int h) 
    { weight = w;
      hungriness = h;
    }
    public void eat (Grass g)
    {
        weight = weight + g.amount;
        hungriness = hungriness - g.amount;
    }
}
```

Anatomy of a Class

```
public class Rabbit
{
    public int weight;
    public int hungriness;
    public Rabbit(int w, int h) 
    { weight = w;
      hungriness = h;
    }
    public void eat (Grass g)
    {
        weight = weight + g.amount;
        hungriness = hungriness - g.amount;
    }
}
```