Java as an OO Language

- Java is considered an OO language
  - Reminder: "Object-Oriented" design generally means "Class-Oriented".
  - Same is true in programming: "OO Programming" is really very much class-oriented
- Java makes OO programming possible, but...
  - You can also write Java programs which violate OO principles

Just because a program is written in Java does NOT mean it is OO!

OO Features of Java

Lecture 02B

Classes in Java

- The class is the basic unit of a Java program

```java
public class MyClass {
    // methods, variables, etc.
}
```

- A .java file typically contains one public class
  - There can be private classes and nested or "inner" classes, too

OO Features of Java

- We look at the most basic OO features of Java
  - Much detail is omitted
- These allow us to implement classes, objects, messages, etc.
- This should be Java you already know!
  - We review it to point out the OO features and terminology
- There are other OO and non-OO features of Java that we will review later
Objects in Java

- Classes are created when the program is designed (written)
- Objects are created when the program runs
- Objects are instances of classes
- The new operator creates an object
  \textit{new MyClass(...)}

- The newly created object has an internal "name" or reference that unique identifies it

References and Objects

- To use a new object later, save its reference
  \texttt{variable = new MyClass(...)};

- The variable now contains a reference which unique identifies the object
- The variable must be declared of an appropriate type (more later), for example
  \texttt{MyClass variable;}
- More than one variable can refer to the same object

Count The Objects

// Program starts, no MyClass objects yet...
\texttt{variable1 = new MyClass(...)};
//now there is one MyClass object
\texttt{variable2 = variable1;}
//now how many objects?
\texttt{variable3 = new MyClass(...)}
//now how many objects?

Objects and References

- Classes, objects, references, and variables are all different things
- This is VERY IMPORTANT to understand.
- Drawing a picture will help
Declaring a Variable

• Variables are declared in Java by giving a type followed by a variable name: 
  \textit{Student xiaoWang};
• Such a variable can later refer to a Student object, old or new

Declaring vs Creating

• A variable is \textit{declared}. This does not create or change any object.
• An object is \textit{created}. This does not create or change any variable.
  \textit{Student xiaoWang = new Student();}
• Three separate operations take place.
  – A new variable is declared. It does not yet refer to any student.
  – A new Student object is created.
  – Finally, the reference to the new object is assigned to the new variable

Messages in Java

• Messages are implemented in Java by "methods"
• The parameters of the message are the parameters of the method

  \textit{class Employee { 
  \hspace{1cm} public void turnAround(int howMany) \};}
  
  The message (method) is "turnAround". The parameter name "howMany". The parameter value is up to the sender of the message.

Commands in Java

• Reminder: commands are messages which do not return a value
• In Java, commands are methods with \textit{void} return type

  \textit{class Employee { 
  \hspace{1cm} public void turnAround(int howMany) \};}
  
  "void" simply means "there is no return value"
Queries in Java

- Reminder: queries are messages which return a value.
- In Java, commands are methods with any non-void return type.

```java
class Employee {
    public String getMyName() {
        ...
        return somethingOrOther;
    }
}
```

In such a method there will always be at least one return statement with a value.

Sending a Message

- Reminder: to send a message, you must know the name of the object.
- In Java, to send a message, you must have a reference to the object. Then you send the message using this syntax:

```java
variable.methodName(parameters)
```

- We say that the method is "called" or "invoked" on the object.

What's Wrong Here?

- Employee emp;
- emp.turnAround(3);

Saving Return Values

- "A query invocation produces a value."
- This is a fancy way of saying "if you call a method with a non-void return type, it will return a value".

```java
Student john;
john.getMyName();
```

- This is legal, but... the returned value is lost

```java
String age = john.getMyAge(); //save the value
```
Object Attributes

- In Java, attributes (properties) of an object are "instance variables"
- Each object of the class has the same instance variables
- Each object of the class has its own values for the instance variables

```
class Student {
    int age;
    String name;
    ...
}
```

Using Instance Variables

- Instance variables are "persistent"
  - Keep their values even when between messages to the object
- An object can always see and modify its own instance variables
- Can one object see or modify the instance variables of another object?
  - Yes, you CAN program that way in Java
  - It is not considered good OO style!

Constructors

- A constructor is a special type of method
- Invoked by the new operator when an object is created
- The constructor "initializes" the object
- Constructor is neither a command or a query
- A constructor can never be invoked except when the object is created
- Examples later

```
class Employee {
    String name; //constructor should initialize
    String company = "IBM"; //already initialized
    ...
}
```
Summary

• Classes are the basic unit of Java program design
• Objects are created by `new`
• Classes, objects, references, and variables are all different things
• Messages correspond to methods
• Parameters and return values of messages correspond to those of methods
• Attributes correspond to instance variables
• Constructors create new objects