Object & Class

Object: contains variables (fields, instance variables) and methods
- Variables: "state" or "characteristics"
  e.g., name, age
- Methods: "behavior" or "action"
  e.g., yell, bounce

Class: blueprint (definition) of an object
- No memory space is reserved for object data
- An object is an instance of a class

Object & Class—an analogy

Object: a folder that stores information (data and instructions)

Class: a drawer in a filing cabinet that holds folders of the same type

What is in an object?

Fields to store data
- score
- addBonus(int)
- getName()

Instructions for dealing with the object
- Reference name (a unique ID of the folder)
- Class name (Drawer name)

Methods

Function: a value is returned at the end of the method

Procedure: the method performs some operations but no value is returned at the end of the method

Using predefined classes

All the predefined classes are collectively called the Java API
- Classes are grouped into packages. E.g., java.io, java.net, javax.swing
- Use the import statement:
  import javax.swing.*;
- To find out what the classes do, read the API specifications:
  http://java.sun.com/api/index.html
Class **JFrame**

- Deals with windows (frames) on the monitor
- See Fig 1.6 in text to see a short list of methods
- See the API documentation for complete specifications

Creating an object

The expression

```
new JFrame()
```

- Creates a `JFrame` object (folder) and gives it a reference name
- Calls method `JFrame()` to set initial values for the object
- Yields the reference of the object

Creating an object

```
JFrame f = new JFrame();
```

Reference variable

- Use a reference variable to hold on to an object:
  
  ```
  JFrame f= new JFrame();
  ```

  Use the class name as a type

Syntax:

```
referenceVariableName . methodName (arguments )
```

Accessing a field

Syntax:

```
referenceVariableName . fieldName
```

Calling instance methods

```
JFrame f= new JFrame();
f . show();
f . setSize(600,200);
int w = f . getWidth();
```

Syntax:

```
referenceVariableName . methodName (arguments )
```