Objects

- An object has:
  - state - descriptive characteristics
  - behaviors - what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

Classes

- An object is defined by a class
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment of that concept
- Multiple objects can be created from the same class
Objects and Classes

A class (the concept)

Bank Account

John's Bank Account
Balance: $5,257

Bill's Bank Account
Balance: $1,245,069

Mary's Bank Account
Balance: $16,833

An object (the realization)

Multiple objects from the same class

Inheritance

• One class can be used to derive another via inheritance
• Classes can be organized into hierarchies

Bank Account

Account

Charge Account

Bank Account

Savings Account

Checking Account

Classes

• A class can contain data declarations and method declarations

int size, weight;
char category;

Data declarations

Method declarations

Bank Account Example

acct1
acctNumber 72354
balance 102.56
name

"Ted Murphy"

acct2
acctNumber 69713
balance 40.00
name

"Jane Smith"

Creating Objects

• A variable holds either a primitive type or a reference to an object
• A class name can be used as a type to declare an object reference variable

String title;

• No object is created with this declaration
• An object reference variable holds the address of an object
• The object itself must be created separately

Creating Objects

• Generally, we use the new operator to create an object

  title = new String ("Java Software Solutions");

  This calls the String constructor, which is a special method that sets up the object

  • Creating an object is called instantiation
  • An object is an instance of a particular class
Invoking Methods

- We've seen that once an object has been instantiated, we can use the `dot operator` to invoke its methods.
  
  \[ \text{count} = \text{title.length()} \]

- A method may return a value, which can be used in an assignment or expression.

- A method invocation can be thought of as asking an object to perform a service.

References

- Note that a primitive variable contains the value itself, but an object variable contains the address of the object.

- An object reference can be thought of as a pointer to the location of the object.

- Rather than dealing with arbitrary addresses, we often depict a reference graphically:

  \[
  \begin{align*}
  \text{num1} & \rightarrow 38 \\
  \text{name1} & \rightarrow \text{"Steve Jobs"}
  \end{align*}
  \]

Assignment Revisited

- The act of assignment takes a copy of a value and stores it in a variable.

- For primitive types:

  Before:
  \[
  \begin{align*}
  \text{num1} & \rightarrow 38 \\
  \text{num2} & \rightarrow 96
  \end{align*}
  \]

  \[ \text{num2} = \text{num1}; \]

  After:
  \[
  \begin{align*}
  \text{num1} & \rightarrow 38 \\
  \text{num2} & \rightarrow 38
  \end{align*}
  \]

Reference Assignment

- For object references, assignment copies the address:

  Before:
  \[
  \begin{align*}
  \text{name1} & \rightarrow \text{"Steve Jobs"} \\
  \text{name2} & \rightarrow \text{"Steve Wozniak"}
  \end{align*}
  \]

  \[ \text{name2} = \text{name1}; \]

  After:
  \[
  \begin{align*}
  \text{name1} & \rightarrow \text{"Steve Jobs"} \\
  \text{name2} & \rightarrow \text{"Steve Jobs"}
  \end{align*}
  \]

Aliases

- Two or more references that refer to the same object are called aliases of each other.

- That creates an interesting situation: one object can be accessed using multiple reference variables.

- Aliases can be useful, but should be managed carefully.

- Changing an object through one reference changes it for all of its aliases, because there is really only one object.

Garbage Collection

- When an object no longer has any valid references to it, it can no longer be accessed by the program.

- The object is useless, and therefore is called garbage.

- Java performs automatic garbage collection periodically, returning an object's memory to the system for future use.

- In other languages, the programmer is responsible for performing garbage collection.
Storing an int

- int (32 bits = 4 bytes)

Reference Variables

- H

Other Material from Sec 3.2

- Other Material from Sec 3.2

The String Class

- Because strings are so common, we don’t have to use the new operator to create a String object
  - title = "Java Software Solutions";
- This is special syntax that works only for strings
- Each string literal (enclosed in double quotes) represents a String object

String Methods

- Once a String object has been created, neither its value nor its length can be changed
- Thus we say that an object of the String class is immutable
- However, several methods of the String class return new String objects that are modified versions of the original
- See the list of String methods on page 119 and in Appendix M

String Indexes

- It is occasionally helpful to refer to a particular character within a string
- This can be done by specifying the character’s numeric index
- The indexes begin at zero in each string
- In the string “Hello”, the character ‘e’ is at index 0 and the ‘o’ is at index 4
- See StringMutation.java (page 120)
Class Libraries

- A class library is a collection of classes that we can use when developing programs.
- The Java standard class library is part of any Java development environment.
- Its classes are not part of the Java language per se, but we rely on them heavily.
- Various classes we’ve already used (System, Scanner, String) are part of the Java standard class library.
- Other class libraries can be obtained through third-party vendors, or you can create them yourself.

Packages

- The classes of the Java standard class library are organized into packages.
- Some of the packages in the standard class library are:

<table>
<thead>
<tr>
<th>Package</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>java.lang</td>
<td>General support</td>
</tr>
<tr>
<td>java.applet</td>
<td>Creating applets for the web</td>
</tr>
<tr>
<td>java.awt</td>
<td>Graphics and graphical user interfaces</td>
</tr>
<tr>
<td>java.swing</td>
<td>Additional graphics capabilities</td>
</tr>
<tr>
<td>java.net</td>
<td>Network communication</td>
</tr>
<tr>
<td>java.util</td>
<td>Utilities</td>
</tr>
<tr>
<td>javax.xml.parsers</td>
<td>XML document processing</td>
</tr>
</tbody>
</table>
The import Declaration

- All classes of the `java.lang` package are imported automatically into all programs
- It's as if all programs contain the following line:
  ```java
  import java.lang.*;
  ```
- That's why we didn't have to import the `System` or `String` classes explicitly in earlier programs
- The `Scanner` class, on the other hand, is part of the `java.util` package, and therefore must be imported

Where are the packages located?

- C:\Program Files\Java\jdk1.5.0\src.zip
- The zip file contains all libraries that ship with the Java language.

Can you add new packages?

Create a directory `c:\some_path\ISU`

In that directory save the file `Cyclone.java`

At the top of `Cyclone.java` put:

```java
package ISU;
```

Compile 'Cyclone.java' but don't run it.

Set your CLASSPATH to `c:\some_path\`

How to use it?

Put this line at the top of the file that uses your new package:

```java
import ISU.Cyclone;
```

Cyclone.java

```java
package ISU;
public class Cyclone {
  private String msg;
  public Cyclone (String message) {
    msg = message;
  }
  public void printMessage () {
    System.out.println(msg);
  }
}
```

TestCyclone.java

```java
import ISU.Cyclone;
public class TestCyclone {
  public static void main(String[] args) {
    Cyclone cy = new Cyclone("Go Cyclones!");
    cy.printMessage();
  }
}
```
THE END