Introduction To Programming

3. Java Programming Basics (contd.)

```
If (learning == true) {
    Lectures[3].Title = “Java Programming Basics (contd.)”;
    Lectures[3].Commence();
}
```

3. Java Programming Basics (contd.)

3.1 Objectives

- Identify the components of Java programs
- An initial look at OOP (Object Oriented Programming) theory
- Program Design
- Doodle example

3.2 Program Components

- Program Components
  - A Java program is composed of comments, import statements and class declarations
  - A *comment* is any sequence of text that begins with a marker /* and ends with a marker */
  - Another marker for a comment is doubles slashes //
  - The *import* statement allows us to use class that have been written by someone else e.g javabook
  - Every program must include at least one *class*
  - To define a new class we must declare it in the program

3.2 Program Components

- Program Components
  - The syntax for declaring the class is

```
class <class name>{
                    <class member declarations>
}
```

  - The word class is a *reserved* word, this means that it can only be used for a specific purpose
  - If we designate a class as the main class, then we must also define a method called main
  - The syntax is

```
<modifiers> <return type> <method name> (<parameters>){
         method body
}
```

3.2 Program Components

- Program Components
  - <modifiers> is a sequence of terms designating different kinds of methods
  - <return type> is the type of data value returned by a method
  - <method name> is the name of a method
  - <parameters> is a sequence of values passed to a method
  - <method body> is a sequence of instructions

```
public static void main (String args[]){
        //Method Body Here
}
```

3.3 OOP – The Theory

- OOP Theory
  - Two most important concepts
    - Classes
    - Objects
  - An *object* can be defined as a thing, both tangible and intangible. An object is an instance of a class.
  - A program written in Object-Oriented style will consist of interacting objects
  - Example of possible objects for a Bank Account program include:
    - Account, Customer and Transaction
  - An object is comprised of data and operations that can be preformed on that data

3.3 OOP – The Theory

- Objects
  - For example, an Account object may consist of data such as account number, owner, date opened, initial balance and current balance
  - Operations could be deposit, transfer and withdrawal
  - Almost all non-trivial programs will have many objects of the same type
  - A program written in object-oriented style will consist of interacting objects

3.3 OOP – The Theory

- Classes
  - A *class* can be described as a template that defines what objects look like
  - An object is an *instance* of a class
  - An object is an instance of exactly one class
  - A class must be defined before you can create an instance (object) of the class

3.4 Program Design

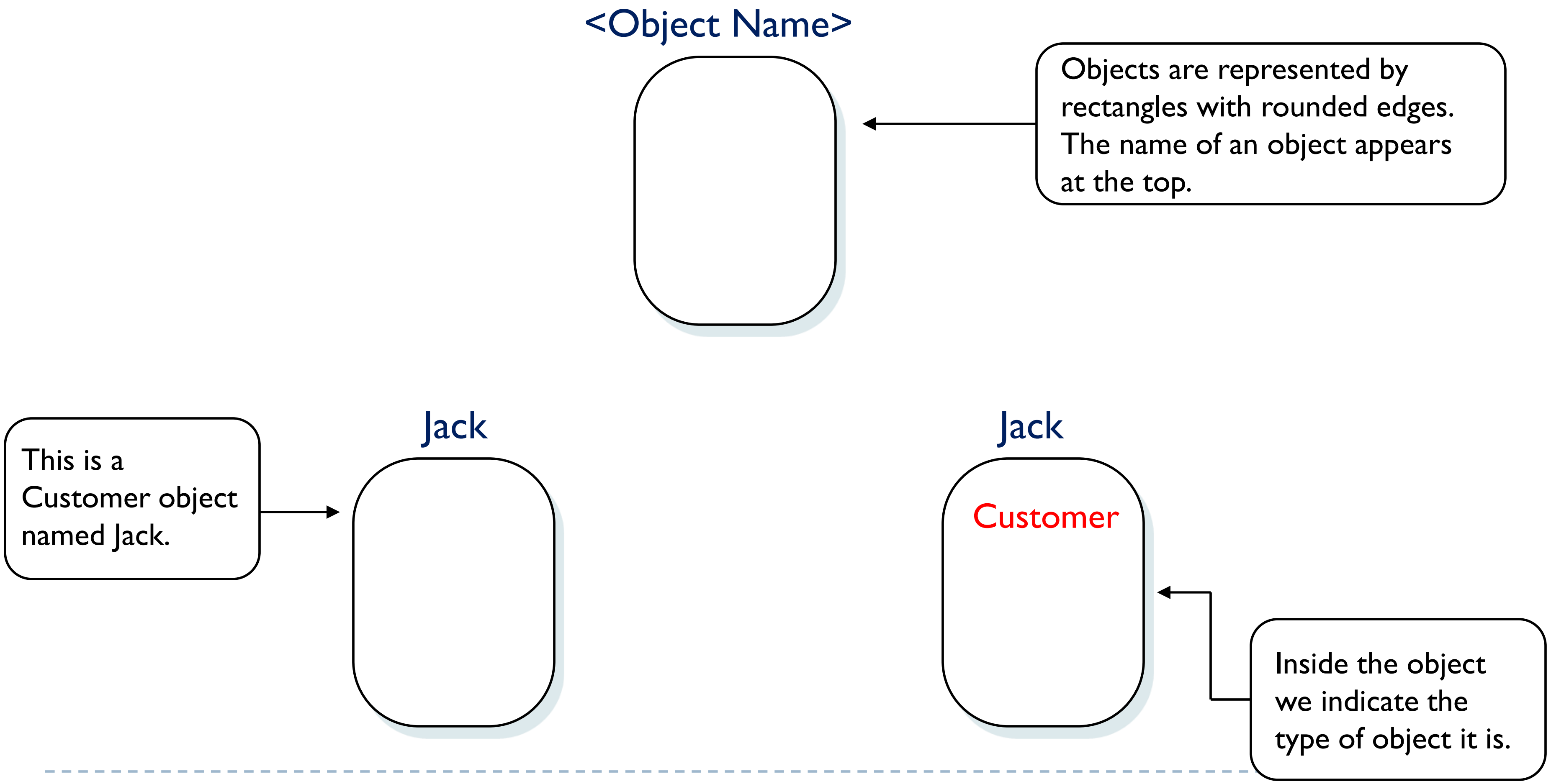
- Program Design
  - To be able to write programs knowing the components of OO programs is not enough.
  - It is important to know the process of developing programs.
  - A very important phase of the Software Development Life Cycle is the design phase.
  - It is important to design a solution to a problem before we begin to write a program.
  - A number of different techniques which can be used:
    - Object diagrams
    - I-P-O diagrams

3.5 Object Diagrams

- Object Diagrams
  - Object diagrams allows us to graphically represent any classes or objects that might be needed in our program.
  - Remember any non-trivial program is going to consist of a number of interacting objects. Therefore it is important that we spend time on the design.
  - Our object diagram will also demonstrate how the objects interact and communicate with each other.

3.5 Object Diagrams

- Graphical Representation of an Object



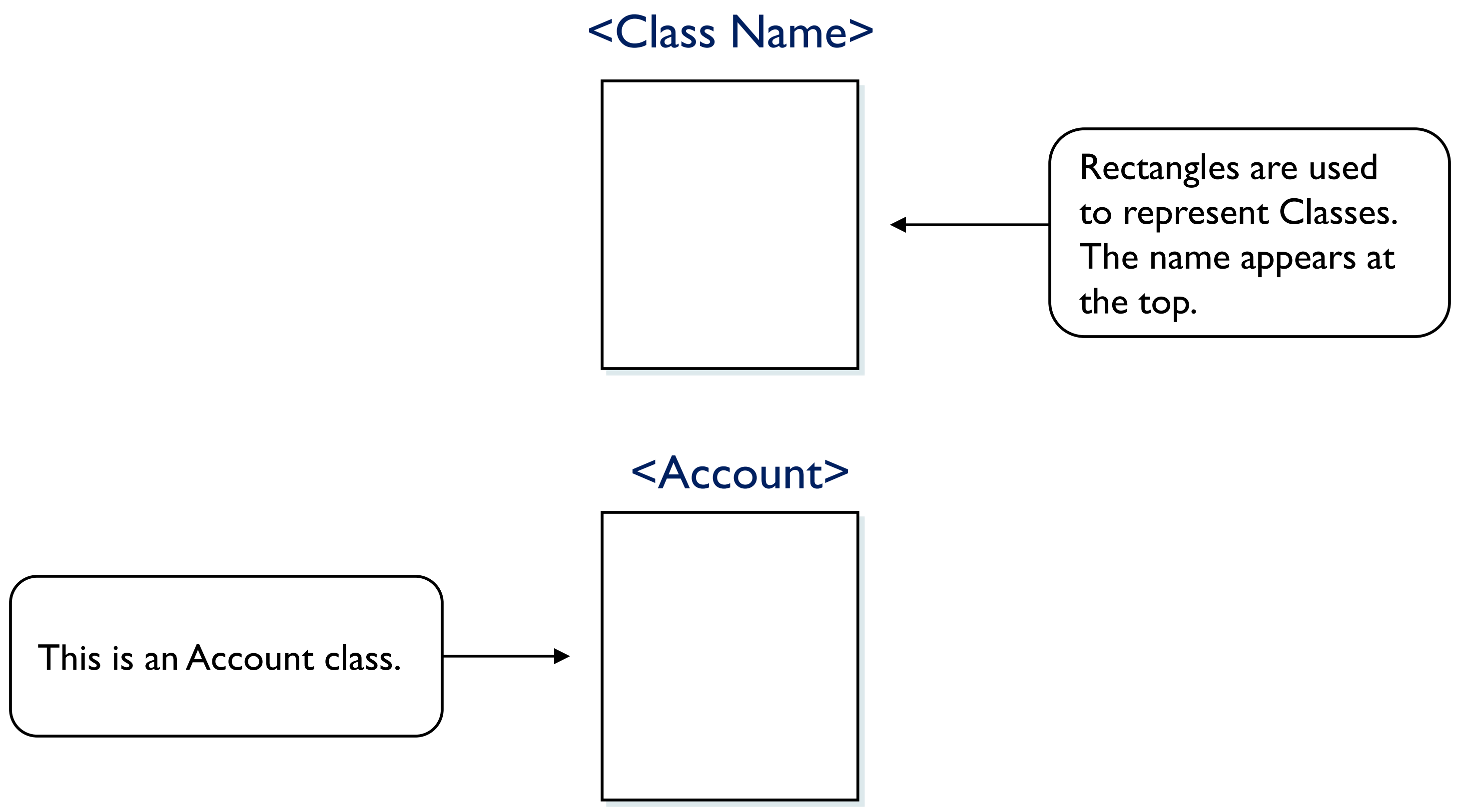
Objects are represented by rectangles with rounded edges. The name of an object appears at the top.

This is a Customer object named Jack.

Inside the object we indicate the type of object it is.

3.5 Object Diagrams

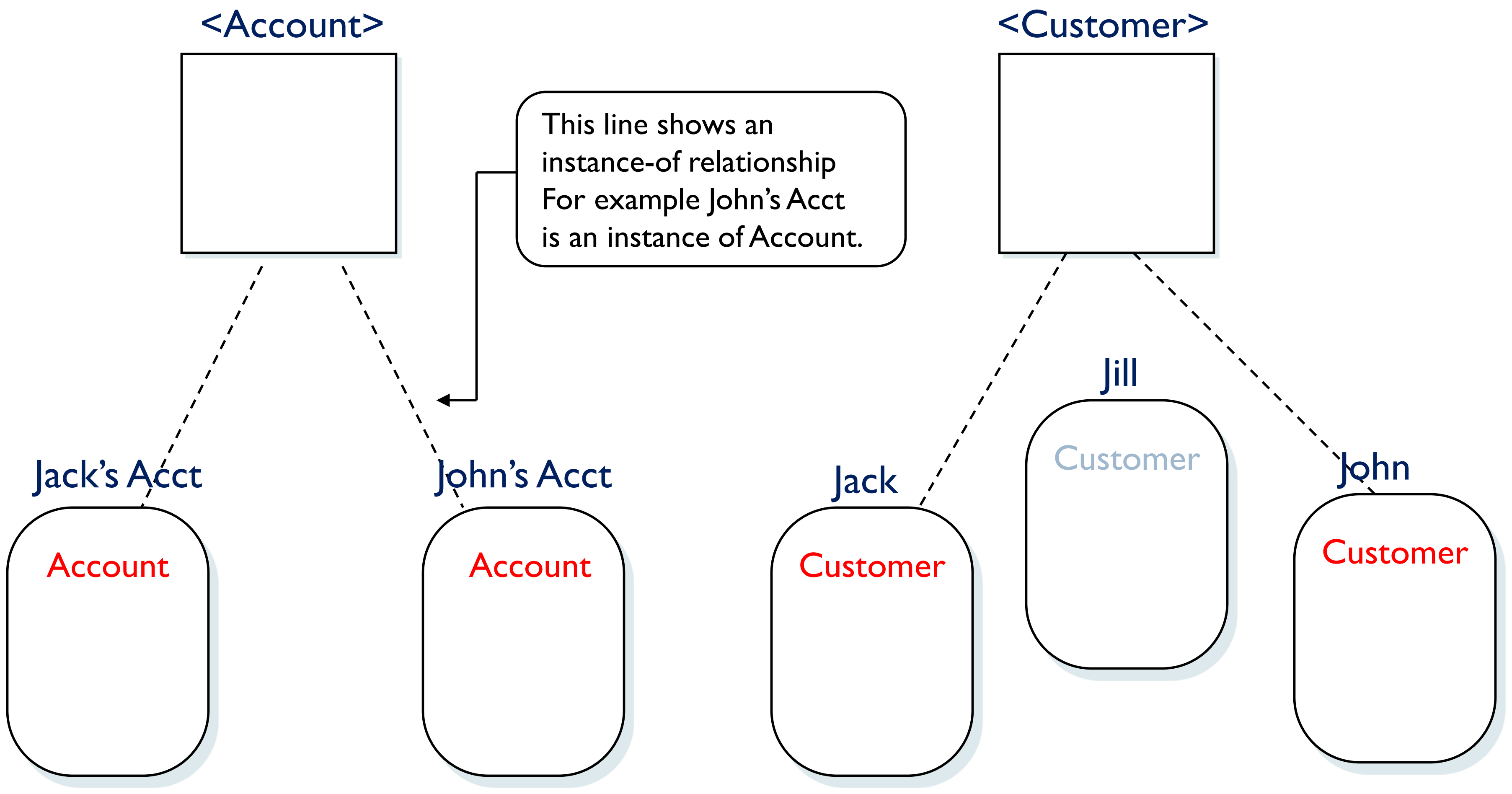
- Graphical Representation of an Class



<Class Name>
Rectangles are used to represent Classes. The name appears at the top.
<Account>
This is an Account class.

3.5 Object Diagrams

- Classes & Objects


<Account>
<Customer>
This line shows an instance-of relationship For example John's Acct is an instance of Account.
Jack's Acct
Account
John's Acct
Account
Jack
Customer
Jill
Customer
John
Customer

3.6 I-P-O Diagrams

- I-P-O Diagrams

  - An I-P-O diagram stands for Input Process Output
  - In order to develop an application we must know
    - What is the input?
    - What do we need to do with that?
    - What is the output?

  - The process is described using an algorithm
  - An algorithm can be defined as an effective procedure for solving a problem in a finite number of steps.

3.6 I-P-O Diagrams

- Algorithms

  - It is *effective*, which means that an answer is found and it finishes, that is it has a *finite* number of steps.
  - We can think of an algorithm as a set of instructions which we must follow in order to complete a task successfully.
  - An example is a recipe that we must follow when baking a cake.

3.6 I-P-O Diagrams

- Sample Algorithm

```
import javabook

define a class

  this class contains one method – main

        declare an object – mwindow
        create an object – mwindow
        display the object

  end the method

end the class
```

3.6 I-P-O Diagrams

- Sample I-P-O Diagram –Adding Two Numbers

Input	Process	Output
	import javabook	
int num1		
	define a class	int sum
	this class contains one method – main	
int num2	declare an object – mwindow, oBox, iBox	
	create an object – mwindow, oBox, iBox	
	display the object	
	get input – num1, num2	
	compute – sum = num1 + num2	
	output – print sum	
	end the method	
	end the class	

3.7 Doodle

-

- This following application allows you to draw a picture by dragging the mouse

- To draw a picture move the mouse while holding down the left button

- To erase the picture click on the right mouse button

3.7 Doodle

```
/*
 *  Doodle.java
 *
 *  Written on: 1st Sept 08
 */
import javabook.*;

class Doodle{
    public static void main ( String args[] ){
        SketchPad   doodleBoard;
        doodleBoard = new SketchPad();
        doodleBoard.show();
    }
}
```

Don't forget to comment your program!!

This statement creates a new Sketchboard object doodleBoard

This statement makes doodleBoard appear on the screen

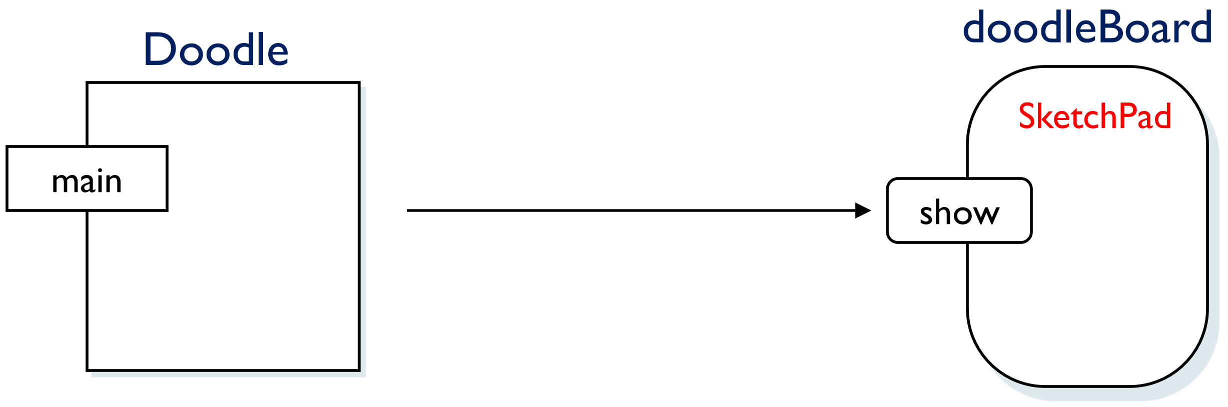
3.7 Doodle

- Explanation

  - The program consists of two classes Doodle and SketchPad
    - the Doodle class is defined in the Doodle program, the Sketchpad class is defined outside of this program
  - This program opens a SketchPad window named doodleBoard and makes it appear on the screen by sending the message *show* to it
  - An object diagram specifies the classes and objects used in a program

3.7 Doodle

- Object Diagram



3.7 Doodle

- Algorithm

```
import javabook

define a class

   this class contains one method - main

      declare an object - doodleBoard
      This is an object of the SketchPad class
      create the doodleBoard object from SketchPad class
      display the object usign the show method


  end the method

end the class
```

3.8 Summary

- An object-oriented program uses objects and classes
- An object is an instance of a class
- A class is a blueprint by which we define objects
- To use an object in a program we first must declare and create an object, then we must send a message to it
- Follow the standard naming convention in writing your java programs to make them easier to read
- Although not required to run the program, comments make code easy-to-understand
- Every program must have at least one class, the main class must contain a method call main