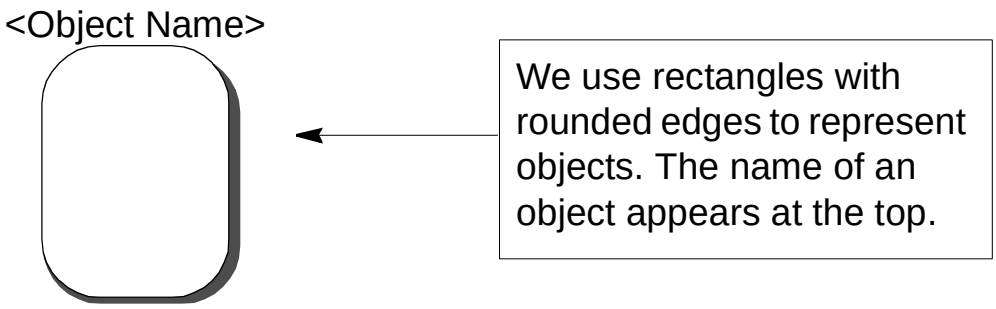
Chapter 1

Introduction to Object-Oriented Programming and Software Development

OBJECTIVES
After you have read and studied this chapter, you should be able to

- Name the basic components of object-oriented programming.
- Differentiate classes and objects.
- Differentiate class and instance methods.
- Differentiate class and instance data values.
- Draw object diagrams using icons for classes, objects, and other components of object-oriented programming.
- Describe the significance of inheritance in object-oriented programs.
- Name and explain the stages of the software life cycle.

FIGURE 1.1 A graphical representation of an object.



Example:

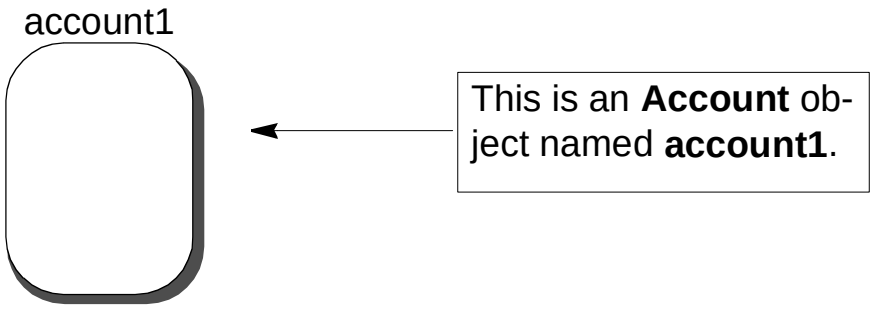


FIGURE 1.2 Two **Customer** objects with the names **Jack** and **Jill** and one **Account** object with the name **SV129**.

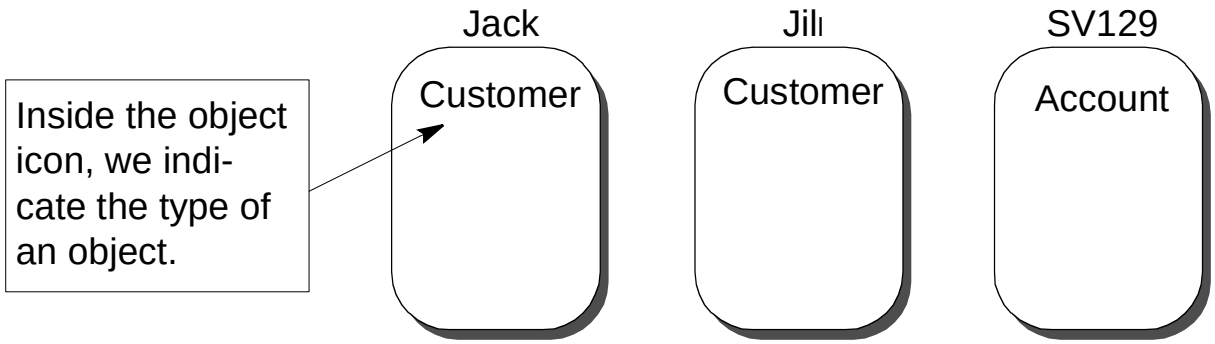


FIGURE 1.3 A graphical representation of a class.

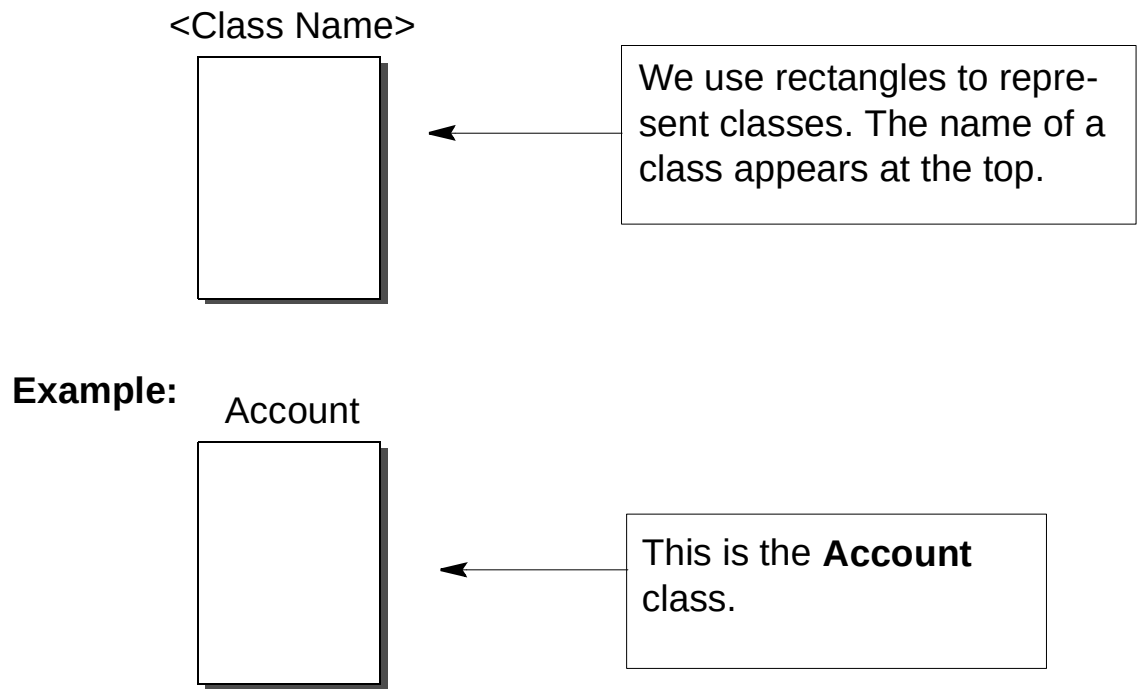


FIGURE 1.4 Two classes, **Account** and **Customer**, with two **Account** objects and three **Customer** objects.

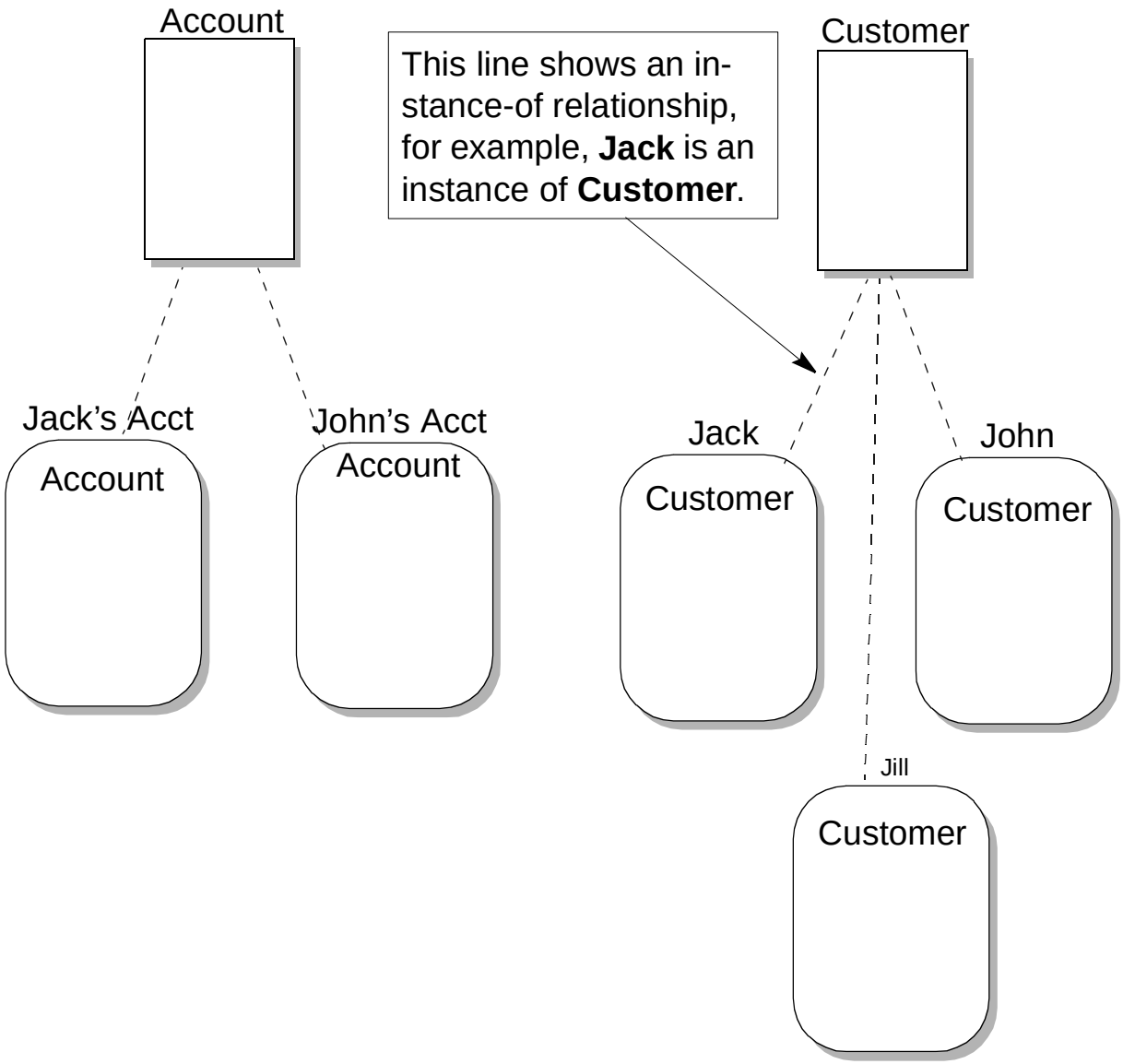


FIGURE 1.5 Sending the message **deposit** to an **Account** object.

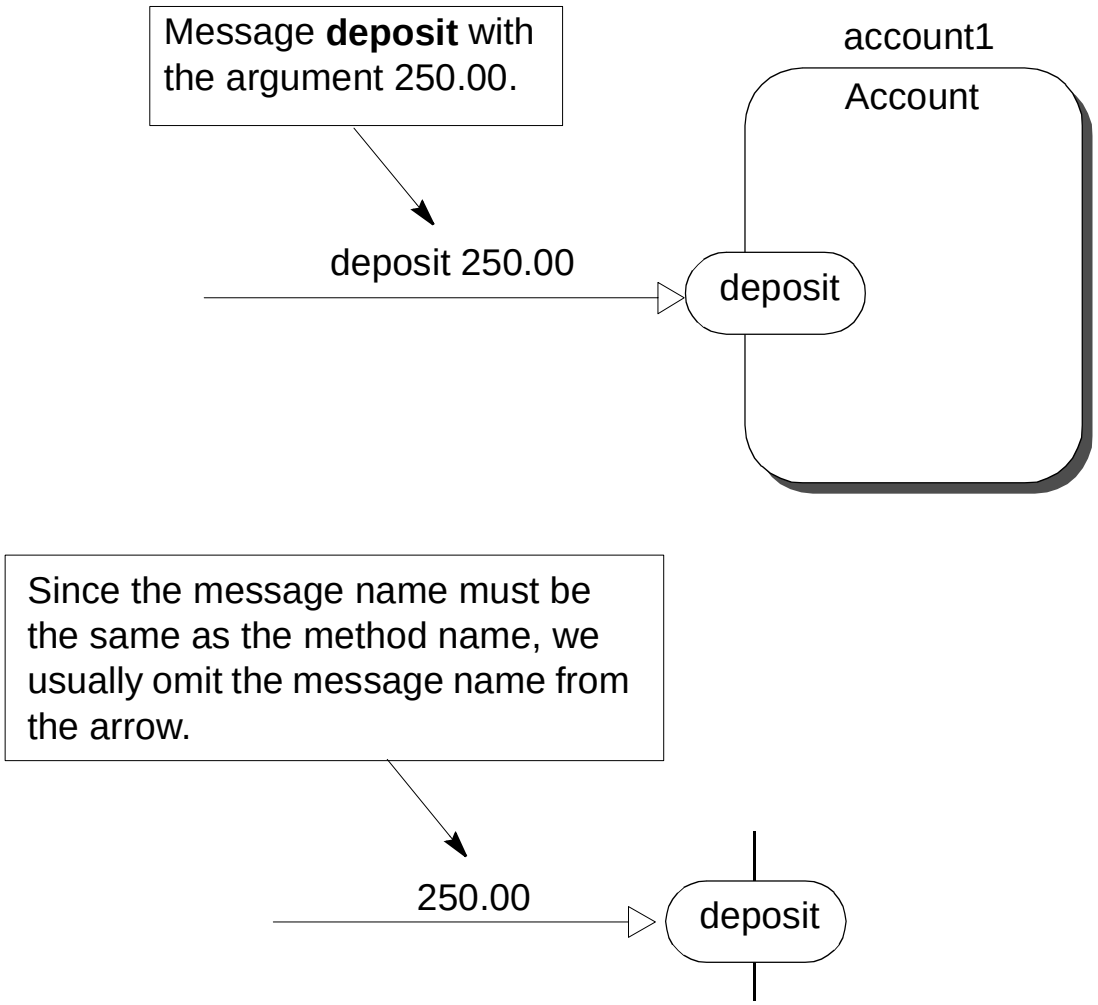


FIGURE 1.6 The result **monthly fee** is returned to the sender of the message.

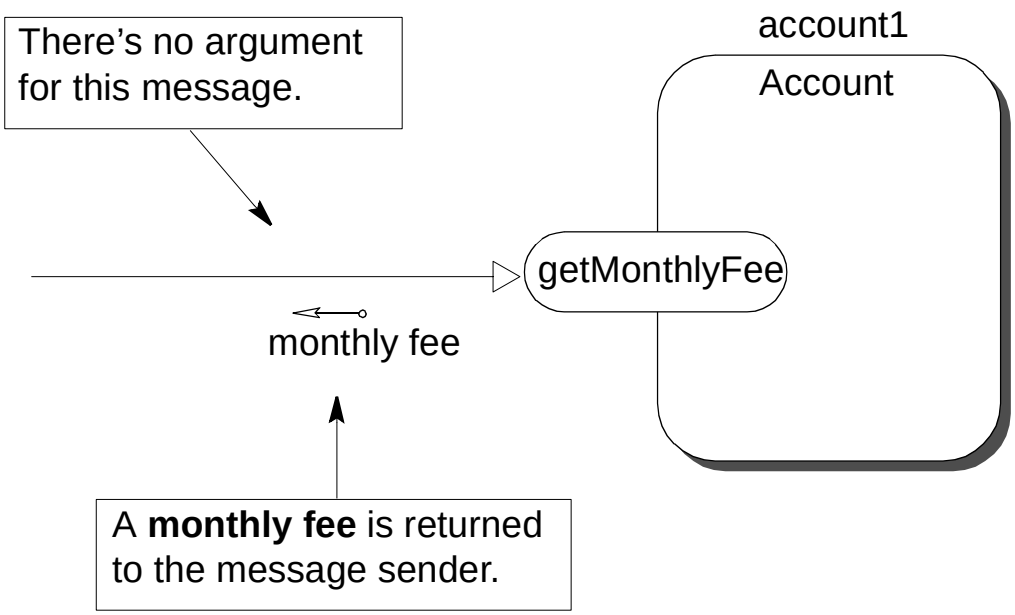


FIGURE 1.7 The newly created **Account** object is returned to the sender of a message.

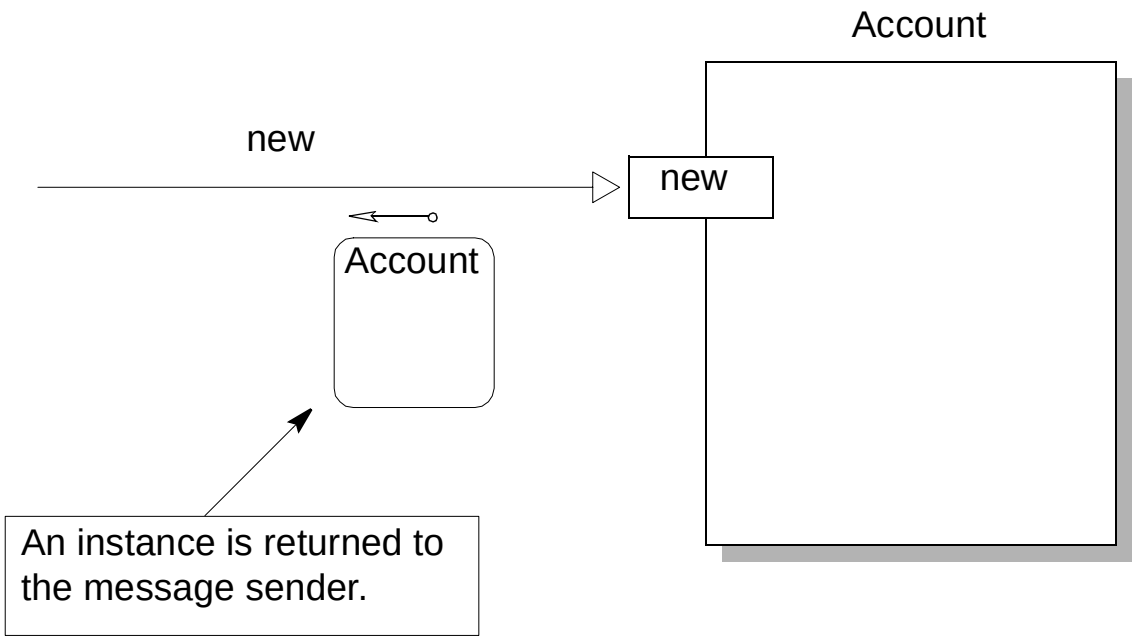


FIGURE 1.8 Graphical representation for classes, instances, and their methods.

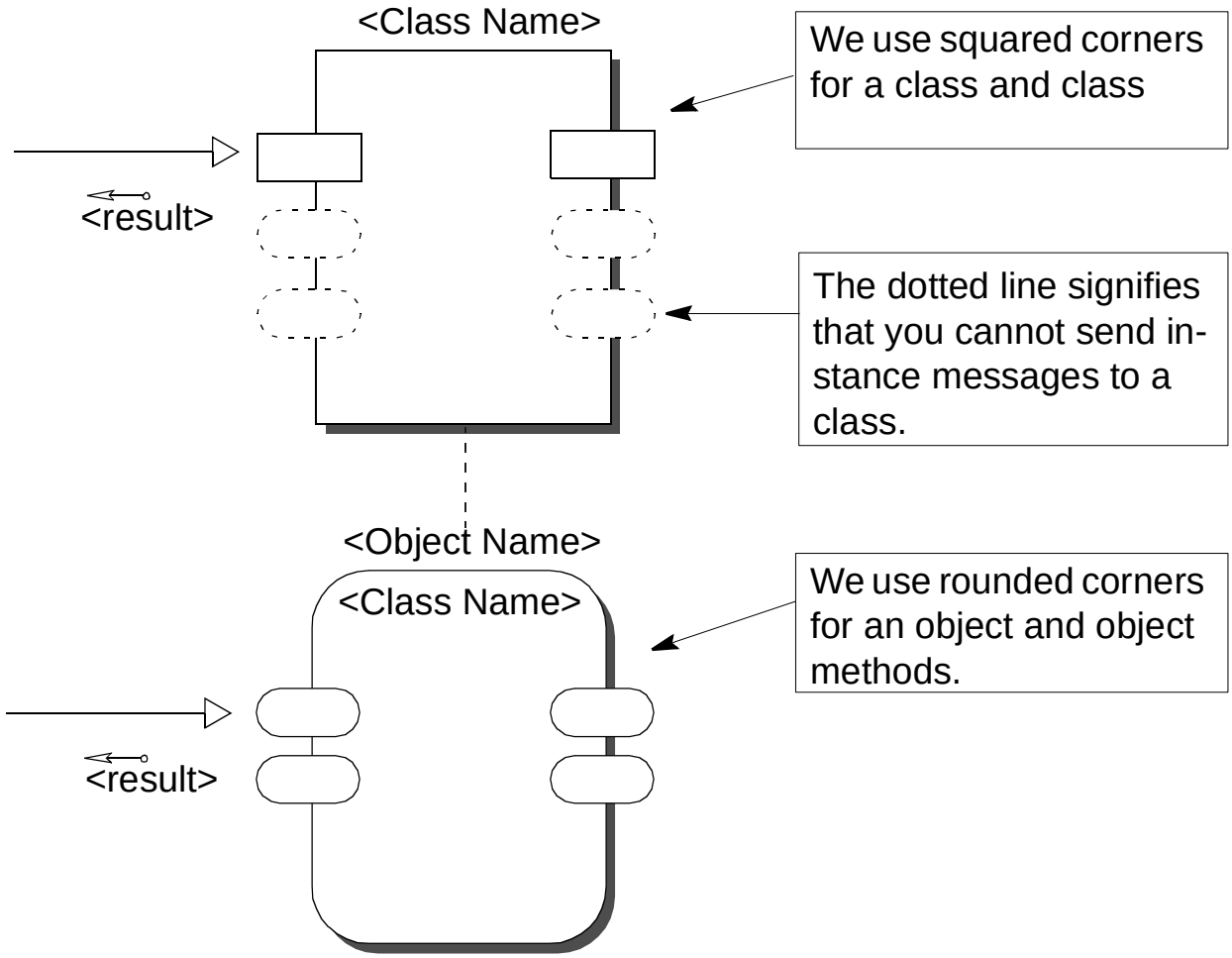


FIGURE 1.9 Three **Account** objects possess the same data value **current balance**, but the actual dollar amounts differ.

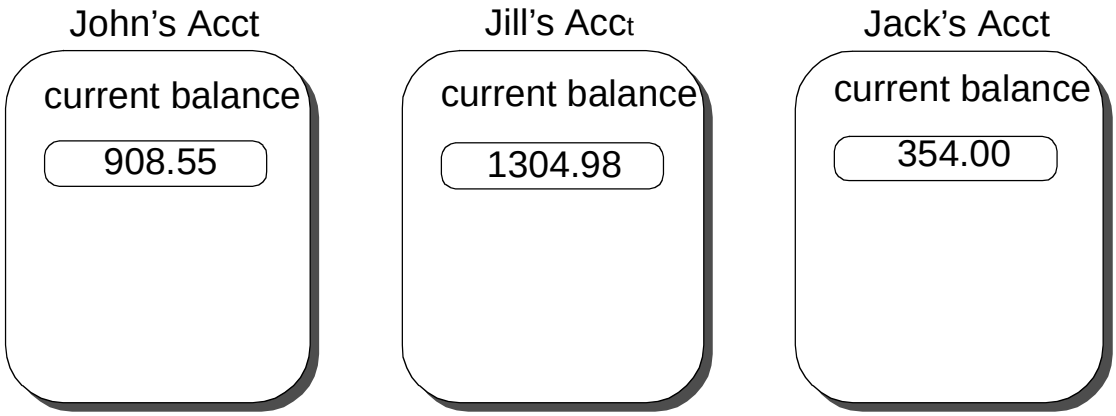


FIGURE 1.10 Three **Account** objects sharing information (**minimum balance** = $100) stored as a class data value.

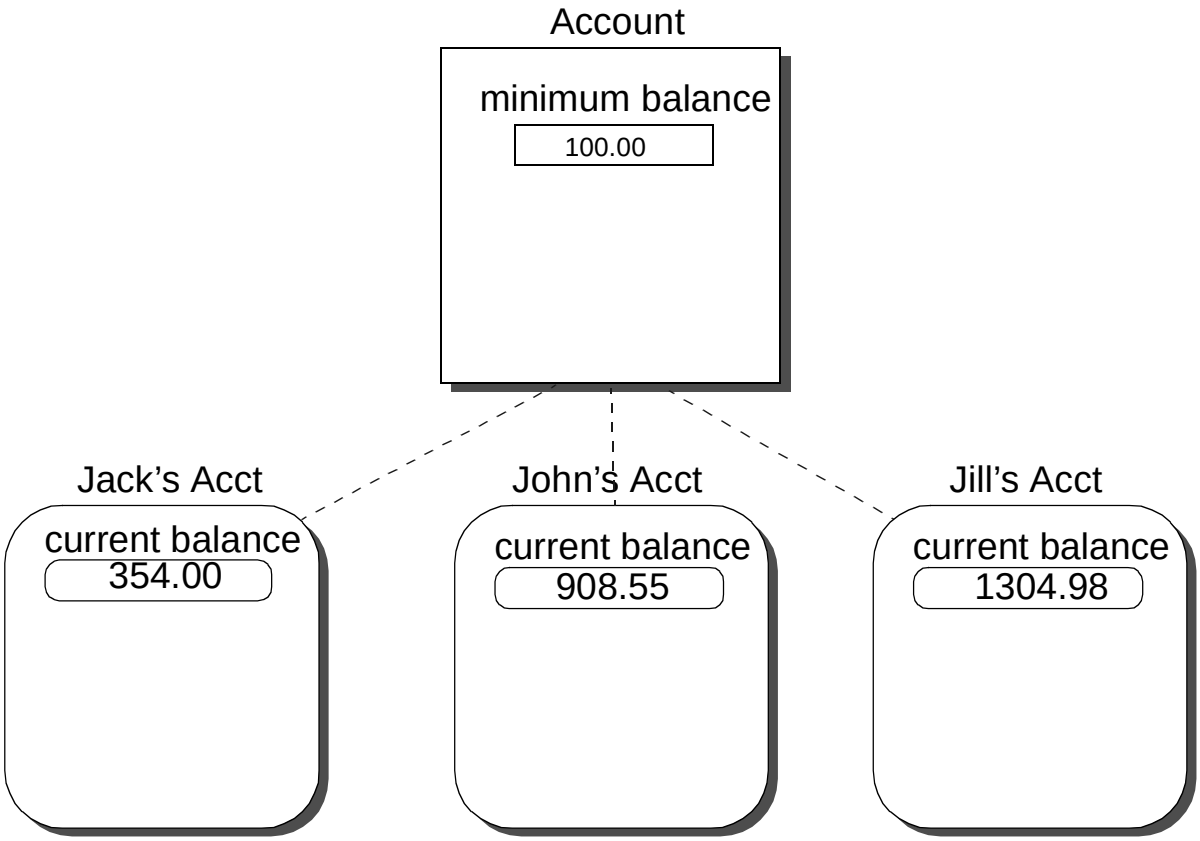


FIGURE 1.11 Three **Account** objects duplicating information (**minimum balance** = $100) in instance data values.

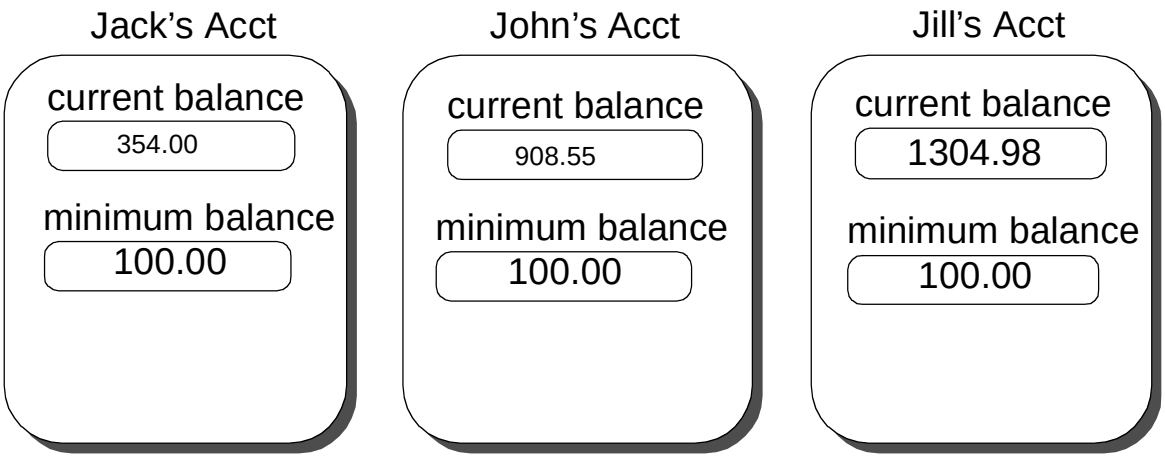


FIGURE 1.12 Graphical representations for four types of data values: class variable, class constant, instance variable, and instance constant.

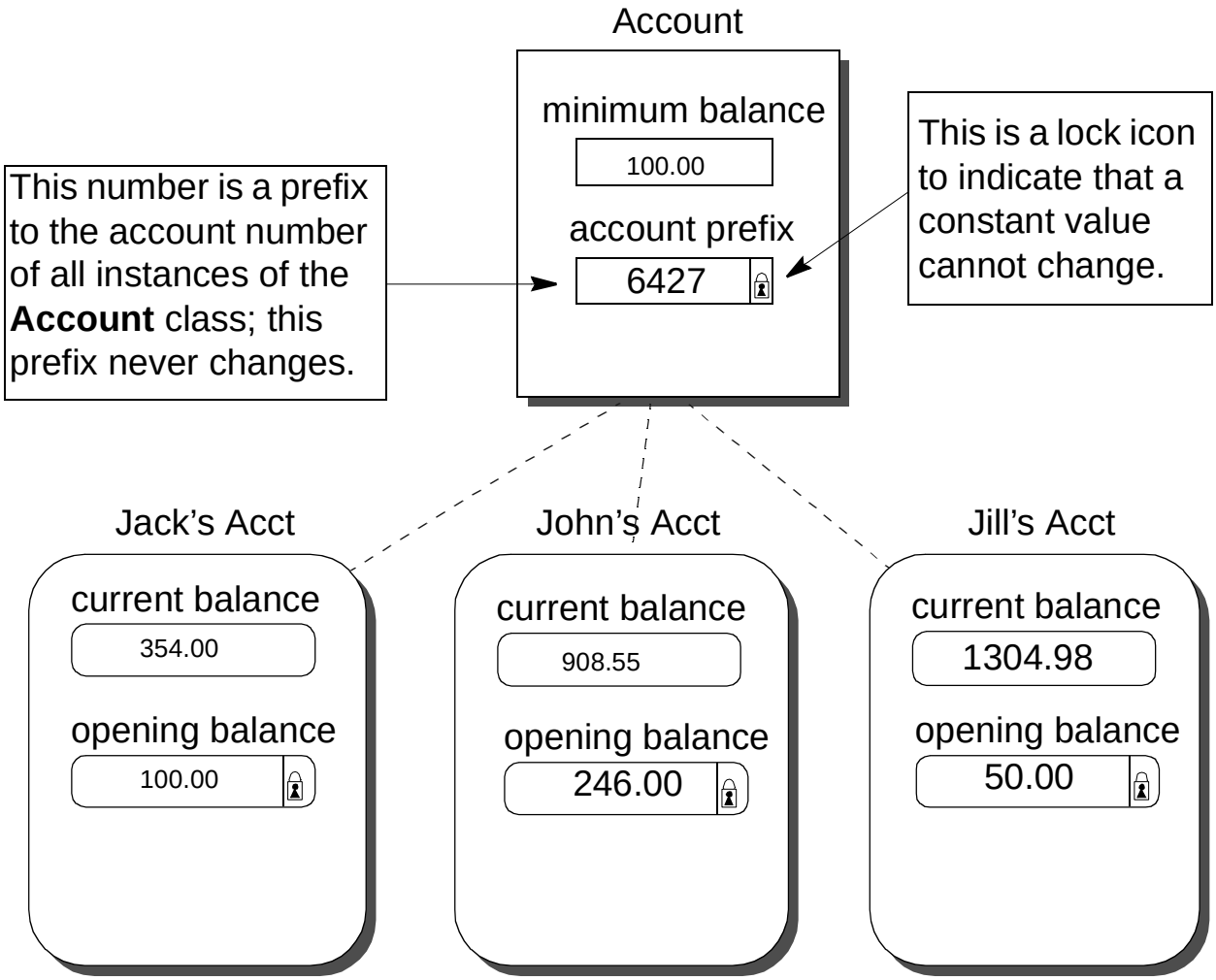


FIGURE 1.13 A superclass **Account** and its subclasses **Savings** and **Checking**.

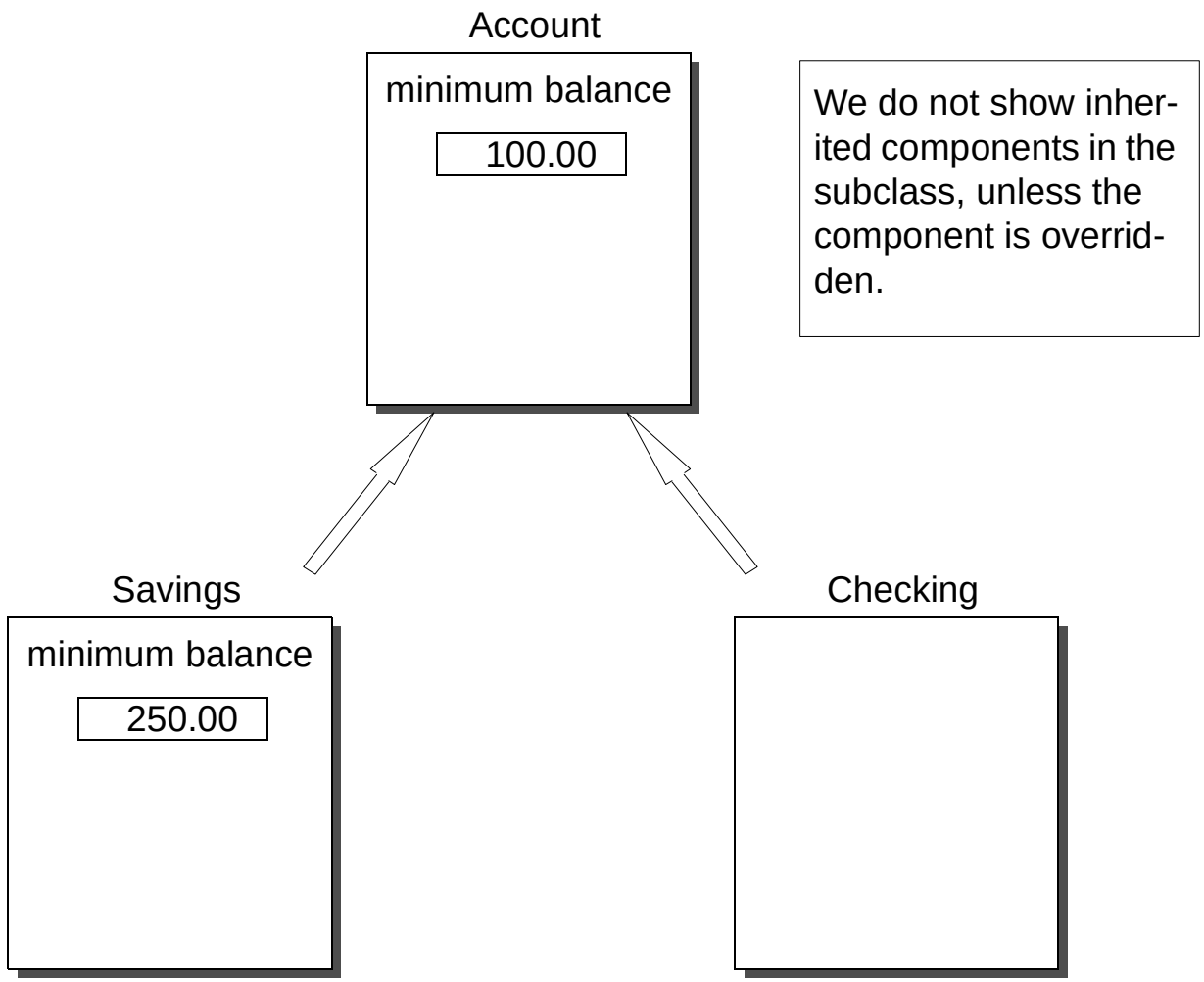
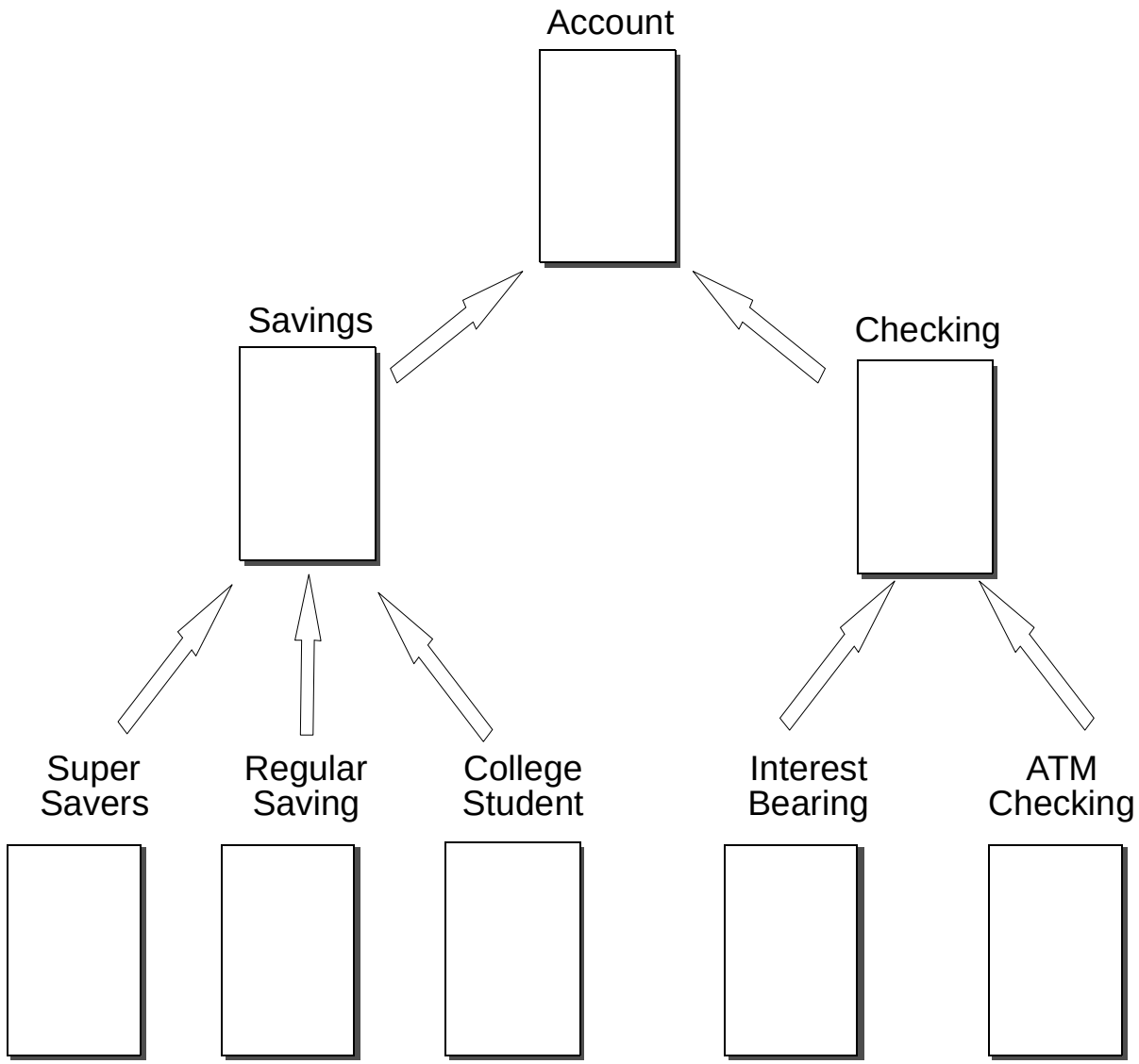


FIGURE 1.14 An example of inheritance hierarchy among different types of accounts.



```
/*
   Program FunTime

   The program will allow you to draw a picture by
   dragging a mouse (move the mouse while holding the left mouse
   button down; hold the button on Mac). To erase the picture and
   start over, click the right mouse button (command-click on Mac).
*/

import javabook.*;

class FunTime
{

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

}
```

This statement makes **doodleBoard** appear on the screen.

This statement creates a new **SketchPad** object **doodleBoard**.

FIGURE 1.15 The window that appears on the screen when the program starts running.

FIGURE 1.16 The same window after a picture is drawn.

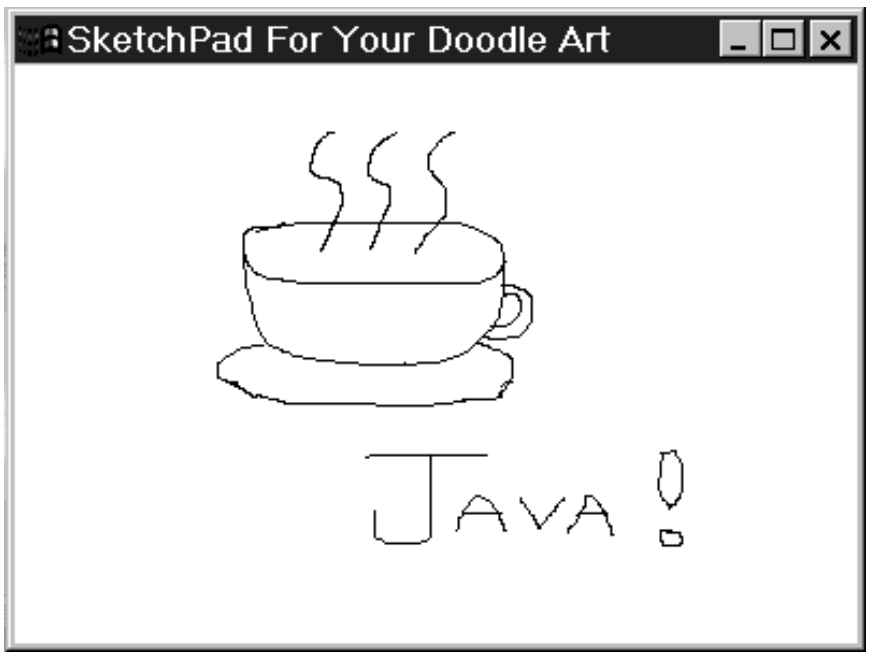


FIGURE 1.17 The object diagram for the **FunTime** program.