**3.5** *(Keyword* new*)* What’s the purpose of keyword new? Explain what happens when you use it.

Answer:

Keyword new requests memory from the system to store an object, then calls the corresponding class’s constructor to initialize the object.

*3.6 (Default Constructors) What is a default constructor? How are an object’s instance variables initialized if a class has only a default constructor?*

Answer:

A constructor can be used to initialize an object of a class when the object is created.

Constructors can specify parameters but cannot specify return types.

A constructor must have the same name as the class

If a class does not define constructors, the compiler provides a default constructor with no parameters, and the class’s instance variables are initialized to their default values.

*3.7 (Instance Variables) Explain the purpose of an instance variable.*

Answer:

When each object of a class maintains its own copy of an attribute, the field that represents the attribute is also known as a(n) instance variable

Variables or methods declared with access modifier private are accessible only to methods of the class in which they’re declared. Declaring instance variables with access modifier private is known as data hiding. Classes often provide public methods to allow the class’s clients to set or get private instance variables

*3.8 (Using Classes Without Importing Them) Most classes need to be imported before they can be used in an application. Why is every application allowed to use classes System and String without first importing them?*

Answer:

Classes System and String are in package java.lang, which is imported implicitly into all source-code files that’s why we don’t need to import them explicitly.

*3.9 (Using a Class Without Importing It) Explain how a program could use class Scanner without importing it.*

Answer:

**import** declarations are not required if you always use fully qualified class names. For example, we can use fully qualified Scanner class as shown below.

java.util.Scanner input = new java.util.Scanner( System.in );

*3.10 (set and get Methods) Explain why a class might provide a set method and a get method for an instance variable.*

Answer:

Declaring instance variables with access modifier private is known as data hiding.

Variables or methods declared with access modifier private are accessible only to methods of the class in which they’re declared. Classes often provide public methods to allow the class’s clients to set or get private instance variables

The names of these methods need not begin with set or get, but this naming convention is recommended and is required for special Java software components called JavaBeans.

*3.11 (Modified GradeBook Class) Modify class GradeBook (Fig. 3.10) as follows:*

*a) Include a String instance variable that represents the name of the course’s instructor.*

*b) Provide a set method to change the instructor’s name and a get method to retrieve it.*

*c) Modify the constructor to specify two parameters—one for the course name and one for the instructor’s name.*

*d) Modify method display Message to output the welcome message and course name, followed by "This course is presented by: " and the instructor’s name.*

*Use your modified class in a test application that demonstrates the class’s new capabilities.*

Answer:

GradeBook.java

package com.ctu.dietel.chapter3;

//GradeBook class with a constructor to initialize the course name and Instructor name

public class GradeBook {

private String courseName; // course name for this GradeBook

private String courseInstructor; // course Instructor name for this GradeBook

// constructor initializes courseName and Instructor name with String arguments

public GradeBook(String name, String courseInstructorName) // constructor name is class name

{

courseName = name; // initializes courseName

courseInstructor = courseInstructorName; // initializes course Instructor name

} // end constructor

// method to set the course name

public void setCourseName(String name) {

this.courseName = name; // store the course name

} // end method setCourseName

// method to retrieve the course name

public String getCourseName() {

return courseName;

} // end method getCourseName

// method to retrieve the course Instructor name

public String getCourseInstructor() {

return courseInstructor;

} // end method getCourseInstructor

// method to set the Course Instructor name

public void setCourseInstructor(String courseInstructor) {

this.courseInstructor = courseInstructor;

} // end method setCourseInstructor

// display a welcome message to the GradeBook user

public void displayMessage() {

// this statement calls getCourseName to get the name of the course this GradeBook represents

System.out.printf("Welcome to the grade book for %s!\n ", getCourseName());

// this statement calls getCourseInstructor to get the course Instructor name

System.out.println("This course is presented by: " + getCourseInstructor());

} // end method displayMessage

} // end class GradeBook

GradeBookTest.java

package com.ctu.dietel.chapter3;

public class GradeBookTest {

// main method begins program execution

public static void main(String[] args) {

// create GradeBook object

GradeBook gradeBook1 = new GradeBook("CS101 Introduction to Java Programming", "Madhava Bhima");

GradeBook gradeBook2 = new GradeBook("CS102 Data Structures in Java", "Madhava Bhima");

gradeBook1.displayMessage();

gradeBook2.displayMessage();

} // end main

} // end class GradeBookTest

Source files:

**

Git Hub url:

<https://github.com/madhavabhima/Advanced_Programming/blob/master/Assignment2/GradeBook.java>

<https://github.com/madhavabhima/Advanced_Programming/blob/master/Assignment2/GradeBookTest.java>

*3.12 (Modified Account Class)Modify class Account (Fig. 3.13) to provide a method called debit that withdraws money from an Account. Ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the method should print a message indicating "Debit amount exceeded account balance." Modify class AccountTest*

*(Fig. 3.14) to test method debit.*

Answer:

Account.java

package com.ctu.dietel.chapter3;

//Account class with a constructor to validate and

//initialize instance variable balance of type double.

public class Account {

private double balance; // instance variable that stores the balance

// constructor

public Account(double initialBalance) {

// validate that initialBalance is greater than 0.0;

// if it is not, balance is initialized to the default value 0.0

if (initialBalance > 0.0)

balance = initialBalance;

} // end Account constructor

// credit (add) an amount to the account

public void credit(double amount) {

balance = balance + amount; // add amount to balance

} // end method credit

// Debit an amount from the account

public void debit(double amount){

if(amount > balance){

System.out.println("Withdraw amount exceeded account balance");

}else{

System.out.printf("\nWithdraw %.2f from account balance\n\n", amount);

balance = balance - amount;

}

} // end method debit

// return the account balance

public double getBalance() {

return balance; // gives the value of balance to the calling method

} // end method getBalance

} // end class Account

AccountTest.java

//Inputting and outputting floating-point numbers with Account objects.

import java.util.Scanner;

public class AccountTest {

// main method begins execution of Java application

public static void main(String[] args) {

Account account1 = new Account(50.00); // create Account object

Account account2 = new Account(-7.53); // create Account object

// display initial balance of each object

System.out.printf("account1 balance: $%.2f\n", account1.getBalance());

System.out.printf("account2 balance: $%.2f\n\n", account2.getBalance());

// create Scanner to obtain input from command window

Scanner input = new Scanner(System.in);

double withDrawAmount; // Withdraw amount read from user

System.out.println("Enter Withdraw amount for account1: "); // prompt

withDrawAmount = input.nextDouble(); // obtain user input

account1.debit(withDrawAmount); // deducted from account1 balance

// display balances

System.out.printf("account1 balance: $%.2f\n", account1.getBalance());

System.out.printf("account2 balance: $%.2f\n\n", account2.getBalance());

System.out.print("Enter Withdraw amount for account2: "); // prompt

withDrawAmount = input.nextDouble(); // obtain user input

account2.debit(withDrawAmount); // deducted from account2 balance

// display balances

System.out.printf("account1 balance: $%.2f\n", account1.getBalance());

System.out.printf("account2 balance: $%.2f\n", account2.getBalance());

} // end main

} // end class AccountTest

Source files:



Git Hub url:

<https://github.com/madhavabhima/Advanced_Programming/blob/master/Assignment2/Account.java>

<https://github.com/madhavabhima/Advanced_Programming/blob/master/Assignment2/AccountTest.java>