**9.1** (The Rectangle class) Following the example of the Circle class in Section 9.2, design a class named Rectangle to represent a rectangle. The class contains:

(矩形类 Rectangle) 遵照 9.2 节中 Circle 类的例子，设计一个名为 Rectangle 的类表示矩形。 这个类包括：

■ Two double data fields named width and height that specify the width and height of the rectangle. The default values are 1for both width and height

两个名为 width和height的double 型数据域，它们分别表示矩形的宽和高。width 和 height 的默认值都为1。

■ A no-arg constructor that creates a default rectangle.

创建默认矩形的无参构造方法。

■ A constructor that creates a rectangle with the specified width and height.

—个创建 width 和 height 为指定值的矩形的构造方法。

■ A method named getArea()that returns the area of this rectangle.

一个名为 getArea()的方法返回这个矩形的面积。

■ A method named getPerimeter()that returns the perimeter.

一个名为 getPerimeter()的方法返回周长。

Draw the UML diagram for the class and then implement the class. Write a test program that creates two Rectangle objects —one with width 4and height 40 and the other with width 3.5and height 35.9 Display the width, height, area, and perimeter of each rectangle in this order.

画出该类的UML图并实现这个类。编写一个测试程序.创建两个 Rectangle 对象 一个矩形的宽为 4 而高为 40, 另一个矩形的宽为 3.5而高为 35.9。按照这个顺序显示每个矩形的 宽、高、面积和周长。

答：

Rectangle类：Rectangle.java

public class Rectangle{

double width;

double height;

public Rectangle() {

width = 1;

height = 1;

}

public Rectangle(double width,double height) {

this.width = width;

this.height = height;

}

public double getArea() {

return width \* height;

}

public double getPerimeter() {

return 2 \* (width + height);

}

}

Rectangle测试类：RectangleText.java

public class RectangleText{

public static void main(String[] args) {

Rectangle r1 = new Rectangle(4,40);

System.out.printf("矩形的长：%.2f，宽：%.2f，面积：%.2f，周长：%.2f\n"

,r1.width,r1.height,r1.getArea(),r1.getPerimeter());

Rectangle r2 = new Rectangle(3.5,35.9);

System.out.printf("矩形的长：%.2f，宽：%.2f，面积：%.2f，周长：%.2f"

,r2.width,r2.height,r2.getArea(),r2.getPerimeter());

}

}

Rectangle

width: double

height: double

+ Rectangle()

+ Rectangle(width : double, height: double)

+ getArea(): double

+ getPerimeter(): double

**9.7** (The Account class) Design a class named Account that contains:

(账户类 Account)设计一个名为 Account 的类，它包括：

■ A private int data field named id for the account (default 0).

一个名为 id的 int 类型私有数据域（默认值为 0)。

■ A private double data field named balance for the account (default 0).

一个名为 balance 的 double 类型私有数据域（默认值为 0）。

■ A private double data field named annualInterestRate that stores the current interest rate (default 0). Assume all accounts have the same interest rate.

一个名为 annuallnterestRate 的 double 类型私有数据域存储当前利率（默认值为 0)。假

设所有的账户都有相同的利率。

■ A private Date data field named dateCreated that stores the date when the account was created.

一个名为 dateCreated的 Date 类型的私有数据域，存储账户的开户日期。

■ A no-arg constructor that creates a default account.

一个用于创建默认账户的无参构造方法。

■ A constructor that creates an account with the specified id and initial balance.

一个用于创建带特定 id 和初始余额的账户的构造方法。

■ The accessor and mutator methods for id, balance, and annualInterestRate.

id、balance 和 annuallnterstRate 的访问器和修改器。

■ The accessor method for dateCreated.

dateCreated 的访问器。 s

■ A method named getMonthlyInterestRate()that returns the monthly interest rate.

一个名为 getMonthlyInterestRate()的方法，返回月利率。

■ A method named getMonthlyInterest()that returns the monthly interest.

一个名为getMonthlyInterest()的方法，返回每月利息。

■ A method named withdrawthat withdraws a specified amount from the account

一个名为 withDraw 的方法，从账户提取特定数额。

■ A method named deposit that deposits a specified amount to the account.

一个名为 deposit 的方法向账户存储特定数额。

Draw the UML diagram for the class and then implement the class.

画出该类的 UML 图并实现这个类。

(Hint: The method getMonthlyInterest()is to return monthly interest, not the interest rate. Monthly interest is balance \* monthlyInterestRate. monthlyInterestRate is annualInterestRate /12. Note that annualInterestRateis a percentage, e.g., like 4.5%. You need to divide it by 100.)

提示：方法 getMonthlylnterest() 用于返回月利息，而不是利率。月利息是 balance\*monthly -InterestRate。monthlylnterestRate 是 annualInterestRate/12。注意，annualInterestRate

是一个百分教，比如 4.5%。你需要将其除以 100。

Write a test program that creates an Accountobject with an account ID of 1122, a balance of $20,000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2,500, use the depos it method to deposit $3,000, and print the balance, the monthly interest, and the date when this account was created.

编写一个测试程序，创建一个账户ID 为 1122、余额为 20 000 美元、年利率为 4.5%的

Account 对象。使用 withdraw 方法取款 2500 美元，使用 deposit 方法存款 3000 美元，然后打 印余额、月利息以及这个账户的开户日期。

答：

Account类：Account.java

import java.text.SimpleDateFormat;

import java.util.Date;

public class Account {

private int id;

private double balance;// 余额

private double annualInterestRate;// 年利率

private Date dateCreated;// 开户日期

public Account() {

id = 0;

balance = 0;

dateCreated = new Date();

}

public Account(int id, double balance) {

dateCreated = new Date();

this.id = id;

this.balance = balance;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public double getBalance() {

return balance;

}

public void setBalance(double balance) {

this.balance = balance;

}

public double getAnnualInterestRate() {

return annualInterestRate;

}

public void setAnnualInterestRate(double annualInterestRate) {

this.annualInterestRate = annualInterestRate;

}

public String getDateCreated() {

//格式化日期

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

return sdf.format(dateCreated);

}

public double getMonthlyInterestRate() {

return annualInterestRate / 12;

}

public double getMonthlyInterest() {

return getMonthlyInterestRate() \* balance;

}

public boolean withDraw(double money) {

if (balance > money) {

balance -= money;

return true;

} else

return false;

}

public void deposit(double money) {

balance += money;

}

}

Account测试类：Account Text.java

public class AccountText {

public static void main(String[] args) {

Account a1 = new Account(1122,20000);

a1.setAnnualInterestRate(4.5/100);//年利率

a1.withDraw(2500);//取款 2500 美元

a1.deposit(3000);//存款 3000

System.out.printf("余额:%.2f\n月利息:%.2f\n",a1.getBalance(),

a1.getMonthlyInterest());

System.out.println("账户开户日期:"+a1.getDateCreated());

}

}

Account

Id: int

balance: double

annualInterestRate: double

dateCreated: Date

+ Account()

+ Account(id: int, balance: double)

+ setId(id: int): void

+ getId(): int

+ getBalance(): double

+ setBalance(balance: double): void

+ setAnnualInterestRate(annualInterestRate: double): void

+ getDateCreated(): String

+ getMonthlyInterestRate(): double

+ getMonthlyInterest(): double

+ withDraw(money: double): Boolean

+ deposit(money: double): void

**9.8** (The Fan class) Design a class named Fanto represent a fan. The class contains:

(风扇类 Fan)设计一个名为 Fan 的类来表示一个风扇。这个类包括：

■ Three constants named SLOW, MEDIUM, and FAST with the values 1, 2, and 3to denote the fan speed.

三个名为 SLOW、MEDIUM 和 FAST 而值为 1、2 和 3 的常量，表示风扇的速度。

■ A private int data field named speed that specifies the speed of the fan (the default is SLOW).

一个名为 speed 的 int 类型私有数据域，表示风扇的速度（畎认值为 SLOW)。

■ A private boolean data field named on that specifies whether the fan is on (the default is false).

一个名为 on 的 boolean 类型私有数据域. 表示风扇是否打开（默认值为 false）。

■ A private double data field named radius that specifies the radius of the fan (the default is 5).

一个名为 radius 的 double 类型私有数据域，表示风扇的半径（默认值为 5 )。

■ A string data field named color that specifies the color of the fan (the default is blue).

一个名为 color 的 string 类型数据域，表示风扇的颜色（默认值为 blue)。

■ The accessor and mutator methods for all four data fields.

这四个数据域的访问器和修改器。

■ A no-arg constructor that creates a default fan.

一个创建默认风扇的无参构造方法。

■ A method named toString()that returns a string description for the fan. If the fan is on, the method returns the fan speed, color, and radius in one combined string. If the fan is not on, the method returns the fan color and radius along with the string “fan is off” in one combined string. 一个名为 toString（） 的方法返回描述风扇的宇符串。如果风扇是打开的，那么该方法在一个 组合的宇符串中返回风扇的速度、顔色和半径。如果风扇没有打开，该方法就会返回一个由 “ fan is off” 和风扇颜色及半径组合成的字符串。

Draw the UML diagram for the class and then implement the class. Write a test program that creates two Fan objects. Assign maximum speed, radius 10, color yellow, and turn it on to the first object. Assign medium speed, radius 5, color blue, and turn it off to the second object. Display the objects by invoking their toString method.

画出该类的 UML 图并实现这个类。编写一个测试程序，创建两个 Fan 对象。将第一个对象 设置为最大速度、半径为 10、颜色为 yellow、状态为打开。将第二个对象设置为中等速度、半径为 S 、颜色为 blue、状态为关闭。通过调用它们的 toString 方法显示这些对象。

答：

Fan类：Fan.Java

public class Fan {

final static int SLOW = 1;

final static int MEDIUM = 2;

final static int FAST = 3;

private int speed;

private boolean on;

private double radius;

private String color;

public Fan() {

speed = SLOW;

on = false;

radius = 5;

color = "blue";

}

public int getSpeed() {

return speed;

}

public void setSpeed(int speed) {

this.speed = speed;

}

public boolean isOn() {

return on;

}

public void setOn(boolean on) {

this.on = on;

}

public double getRadius() {

return radius;

}

public void setRadius(double radius) {

this.radius = radius;

}

public String getColor() {

return color;

}

public void setColor(String color) {

this.color = color;

}

@Override

public String toString() {

if (on) {

return "speed: "+speed+",color: "+color+",radius: "+radius;

} else {

return "fan is off "+",color: "+color+",radius: "+radius;

}

}

}

Fan测试类：FanText.Java

public class FanText {

public static void main(String[] args) {

Fan f1 = new Fan();

f1.setSpeed(Fan.FAST);

f1.setRadius(10);

f1.setColor("yellow");

f1.setOn(true);

Fan f2 = new Fan();

f2.setSpeed(Fan.MEDIUM);

f2.setRadius(5);

f2.setColor("blue");

f2.setOn(false);

System.out.println(f1.toString());

System.out.println(f2.toString());

}

}

Fan

SLOW: int

MEDIUM: int

FAST: int

speed: int

on: boolean

radius: double

color: String

+ Fan()

+ setSpeed (speed: int): void

+ getSpeed(): int

+ isOn (): boolean

+ setOn(on: boolean): void

+ getRadius (): double

+ setRadius(radius: double): void

+ getColor (): String

+ setColor(color: String): void

+ toString(): String