



Java Lab – Classes

IMPORTANT! Save all your work to a safe location such as oneDrive.

Create a folder for SDPD into which you will save all your work for this module, arranged how you wish. Ideally you should create a folder each week for your lab exercises. Note that you should create a separate file for each exercise.

Exercise 1

Create a class called Square using the code provided below. This class will have one field (length) and a method called setLength(), which allows for one value to be passed:

```
1 public class Square
2 {
3     private int length;
4
5     public void setLength(int len)
6     {
7         length = len;
8     }
9 }
```

Ensure that you save this file with the appropriate name, i.e., *Square.java*. Save your class. (Note that if you try run this class, you will receive an error that there is no main method in this class, that is okay).

You will now need to create a Java program that creates a single Square object called box1, similar to as shown below:

```
1 public class Exercisel
2 {
3     public static void main(String[] args){
4
5         //Create an instance of the Square object, call it box1
6         Square box1 = new Square();
7
8         //Using the setLength method you created, set the length of the box1 object to 10
9         box1.setLength(10);
10
11     }
12 }
```

Save and run your code to make sure that it compiles and runs as expected. Note that is at this stage, there will be no output to the screen. Now you will need to add a getter method , to “get” the value from the square object.

To do this, add a new method to your square class. This should be a method without parameters that returns the *length* field as an integer:

```
1 public class Square
2 {
3     private int length;
4
5     public void setLength(int len)
6     {
7         length = len;
8     }
9
10    public int getLength()
11    {
12        return length;
13    }
14 }
```

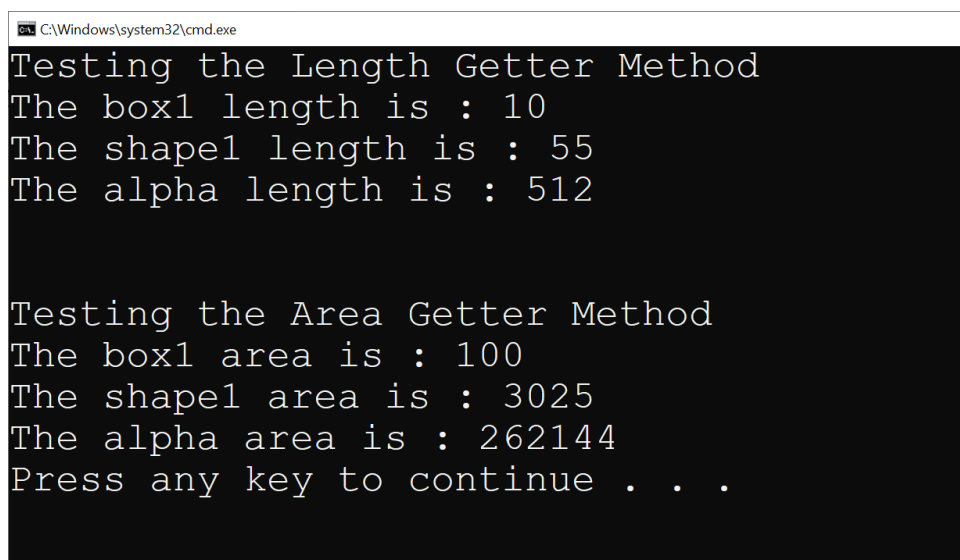
Back in your program, use the `getLength()` method on the `box1` object to get the length value of the object. Ensure you also output this in an appropriate manner so you can see the result (eg, outputting to the console), for example:

```
1 class Exercisel
2 {
3     public static void main(String[] args){
4
5         //Create an instance of the Square object,
6         Square box1 = new Square();
7
8         //Using the setLength method you created,
9         box1.setLength(10);
10
11         System.out.println(box1.getLength());
12
13     }
14 }
```

Amend your program with the following:

- Create a new method called `getArea()` which return the area of the Square ($\text{length} * \text{length}$).
- Create another instance of the Square called `Shape1`. Specify a length of 55 using the `setLength` method, and determine what the area of this is.
- Create a third instance of the Square called `Alpha`. Specify a length of 512 using the `setLength` method, and determine what the area of this is.

Your program should produce output similar to as shown below:



```
C:\Windows\system32\cmd.exe
Testing the Length Getter Method
The box1 length is : 10
The shape1 length is : 55
The alpha length is : 512

Testing the Area Getter Method
The box1 area is : 100
The shape1 area is : 3025
The alpha area is : 262144
Press any key to continue . . .
```

Exercise 2

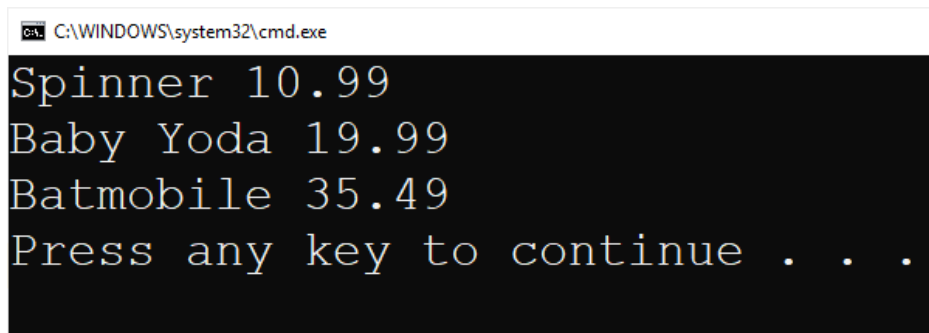
Design a class that holds the name of a toy. The class should be called Toy and have 2 fields – name and price. Create getters and setters for each of these fields. Using this class, create 3 objects in the main method using the following information:

Name: Spinner
Price: 10.99

Name: Baby Yoda
Price: 19.99

Name: Batmobile
Price: 35.49

Output both values for each object using the getters – your output should be similar to as shown below:



```
C:\WINDOWS\system32\cmd.exe
Spinner 10.99
Baby Yoda 19.99
Batmobile 35.49
Press any key to continue . . .
```

Exercise 3

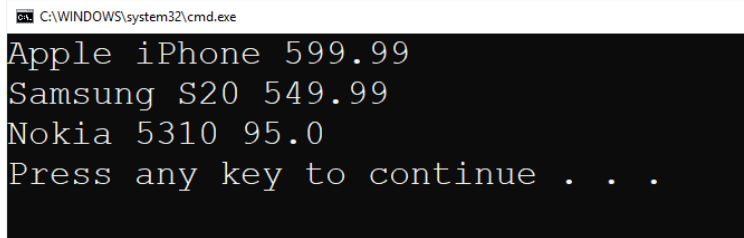
Design a class that holds the name of a mobile phone manufacturer. The class should be called Phone and have 3 fields – module, price and stock. Create getters and setters for each of these fields. Using this class, create 3 objects in the main method using the following information:

Name: Apple iPhone
Price: 599.99
Stock: 11

Name: Samsung S20
Price: 549.45
Stock: 4

Name: Nokia 5310
Price: 95.00
Stock: 6

Output all values for each object using the getters – your output should be similar to as shown below:



```
C:\WINDOWS\system32\cmd.exe
Apple iPhone 599.99
Samsung S20 549.99
Nokia 5310 95.0
Press any key to continue . . .
```

Exercise 4

Design a class that holds the following product data: product name and cost price.

Write appropriate accessor and mutator methods. Include a method called `retailPrice` that calculates the retail price in the shops by adding 45% markup to the cost price. Demonstrate the class by writing a program that creates four instances of it using the information provided below:

Name: Twix Bar

Cost Price: 0.40

Name: Daz Washing Powder 10Kg

Cost Price: 6.05

Name: Colgate Toothpaste

Cost Price: 1.25

Name: Granny Smiths 6-Pack Apples

Cost Price: 1.10

Check your program and class by outputting using the getters – your output should be similar to as shown below:

```
C:\WINDOWS\system32\cmd.exe
Product Details:
Product 1 Name: Twix Bar
Product 1 Cost Price: 0.40
Product 1 Retail Price: 0.58

Product Details:
Product 2 Name: Twix Bar
Product 2 Cost Price: 6.05
Product 2 Retail Price: 8.77

Product Details:
Product 3 Name: Twix Bar
Product 3 Cost Price: 1.25
Product 3 Retail Price: 1.81

Product Details:
Product 4 Name: Twix Bar
Product 4 Cost Price: 1.10
Product 4 Retail Price: 1.60
Press any key to continue . . .
```

Exercise 5

Design a class that holds the following personal data: name, address, age, and phone number.

Write appropriate accessor and mutator methods. Demonstrate the class by writing a program that creates three instances of it using the information provided below:

```
Select C:\Windows\system32\cmd.exe

My information:
Name: Joe Mahoney
Age: 27
Address: 724 22nd Street
Phone: (555) 555-1234

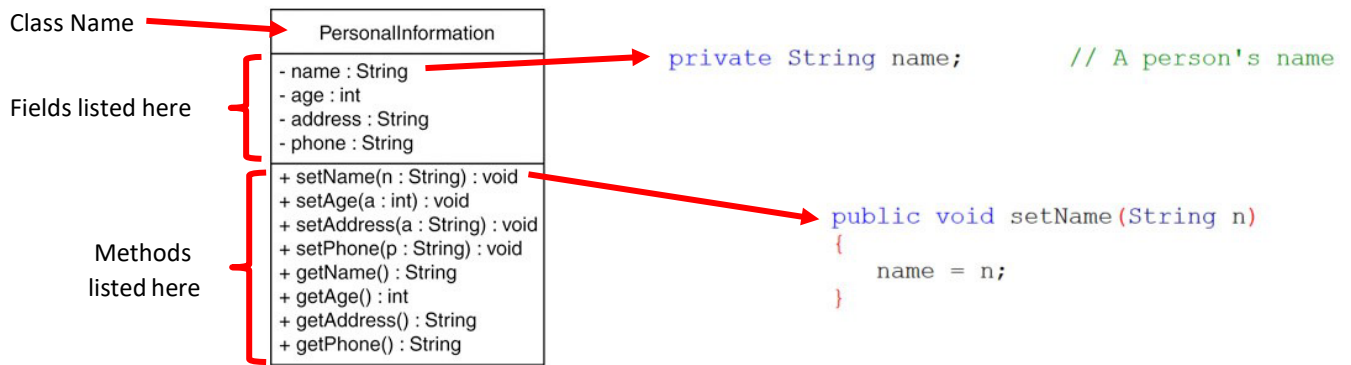
Friend #1's information:
Name: Geri Rose
Age: 24
Address: 149 East Bay Street
Phone: (555) 555-5678

Friend #2's information:
Name: John Carbonni
Age: 28
Address: 22 King Street
Phone: (555) 555-0123
Press any key to continue . . .
```

See the UML diagram below on the classname, fields and methods that are required for this exercise:

PersonallInformation
- name : String - age : int - address : String - phone : String
+ setName(n : String) : void + setAge(a : int) : void + setAddress(a : String) : void + setPhone(p : String) : void + getName() : String + getAge() : int + getAddress() : String + getPhone() : String

UML Diagram Explained:



Exercise 6

Write a class named `Car` that has the following fields:

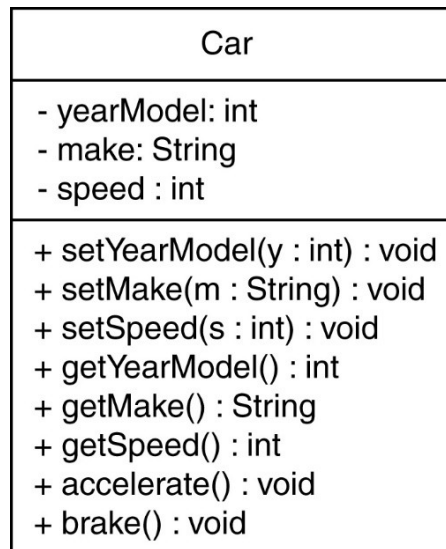
- **yearModel.** The `yearModel` field is an `int` that holds the car's year model.
- **make.** The `make` field references a `String` object that holds the make of the car.
- **speed.** The `speed` field is an `int` that holds the car's current speed.

In addition, the class should have the following constructor and other methods.

- **Accessors.** Appropriate accessor methods should get the values stored in an object's `yearModel`, `make`, and `speed` fields.
- **accelerate.** The `accelerate` method should add 5 to the `speed` field each time it is called.
- **brake.** The `brake` method should subtract 5 from the `speed` field each time it is called.

Demonstrate the class in a program that creates a `Car` object, and then calls the `accelerate` method five times. After each call to the `accelerate` method, get the current speed of the car and display it. Then call the `brake` method five times. After each call to the `brake` method, get the current speed of the car and display it.

The following UML diagram is provided for the exercise:



Your output should look similar to as shown below:

```
C:\Windows\system32\cmd.exe
Current status of the car:
Year model: 2004
Make: Porsche
Speed: 0

Accelerating...
Now the speed is 25

Braking...
Now the speed is 0
Press any key to continue . . . _
```

Exercise 7

Write a class named Employee that has the following fields:

- name. The name field references a String object that holds the employee's name.
- idNumber. The idNumber is an int variable that holds the employee's ID number.
- department. The department field references a String object that holds the name of the department where the employee works.
- position. The position field references a String object that holds the employee's job title.

Write appropriate mutator methods that store values in these fields and accessor methods that return the values in these fields. Once you have written the class, write a separate program that creates three Employee objects to hold the following data:

Name	ID Number	Department	Position
Susan Meyers	47899	Accounting	Vice President
Mark Jones	39119	IT	Programmer
Joy Rogers	81774	Manufacturing	Engineer

The program should store this data in the three objects and then display the data for each employee on the screen.

```
C:\Windows\system32\cmd.exe
Employee #1
Name: Susan Meyers
ID Number: 47899
Department: Accounting
Position: Vice President

Employee #2
Name: Mark Jones
ID Number: 39119
Department: IT
Position: Programmer

Employee #3
Name: Joy Rogers
ID Number: 81774
Department: Manufacturing
Position: Engineer

Press any key to continue . . .
```

Exercise 8

Write a class named *RetailItem* that holds data about an item in a retail store. The class should have the following fields:

- **description.** The description field references a String object that holds a brief description of the item.
- **unitsOnHand.** The unitsOnHand field is an int variable that holds the number of units currently in inventory.
- **price.** The price field is a double that holds the item's retail price.

Write appropriate mutator methods that store values in these fields, and accessor methods that return the values in these fields.

Once you have written the class, write a separate program that creates three Retail Item objects and stores the following data in them:

	Description	Units on Hand	Price
Item #1	Jacket	12	59.95
Item #2	Designer Jeans	40	34.95
Item #3	Shirt	20	24.95

Your output should be similar to as shown below:

```
C:\Windows\system32\cmd.exe
Item #1
Description: Jacket
Units on hand: 12
Price: 59.95

Item #2
Description: Designer Jeans
Units on hand: 40
Price: 34.95

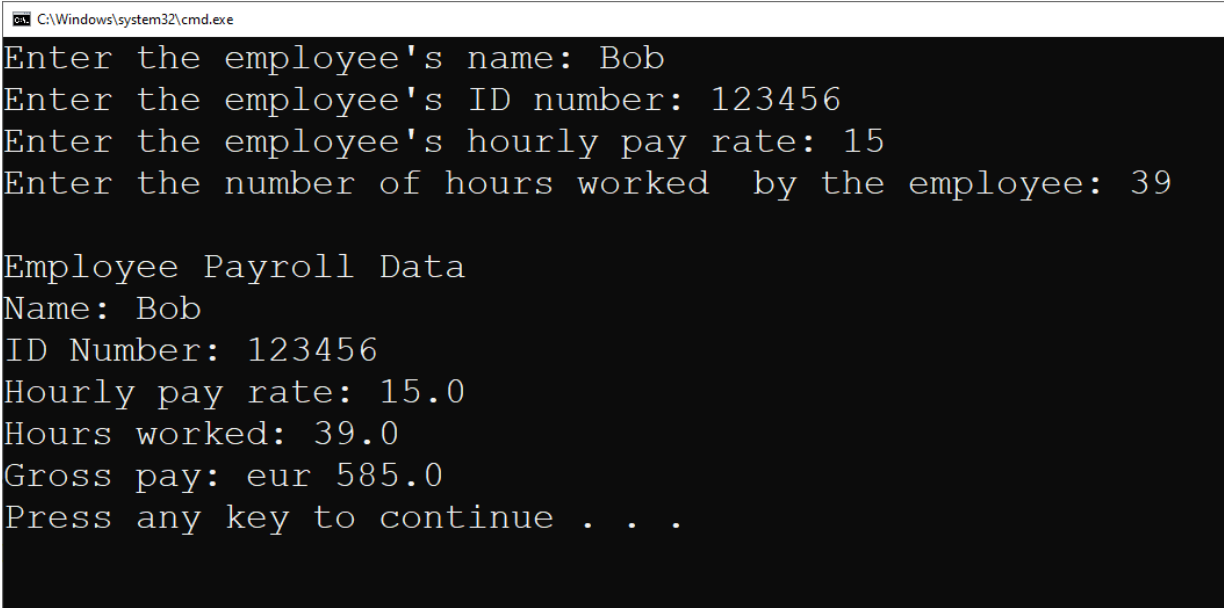
Item #3
Description: Shirt
Units on hand: 20
Price: 24.95
Press any key to continue . . .
```

Exercise 9

Design a **Payroll** class that has fields for an employee's name, ID number, hourly pay rate, and number of hours worked. Write the appropriate accessor and mutator methods that accept the employee's name and ID number as arguments. The class should also have a method that returns the employee's gross pay, which is calculated as the number of hours worked, multiplied by the hourly pay rate.

Write a program that demonstrates the class by creating a Payroll object, then asking the user to enter the data for an employee. The program should display the amount of gross pay earned.

Your program output should be similar to as shown below:



```
C:\Windows\system32\cmd.exe
Enter the employee's name: Bob
Enter the employee's ID number: 123456
Enter the employee's hourly pay rate: 15
Enter the number of hours worked by the employee: 39

Employee Payroll Data
Name: Bob
ID Number: 123456
Hourly pay rate: 15.0
Hours worked: 39.0
Gross pay: eur 585.0
Press any key to continue . . .
```

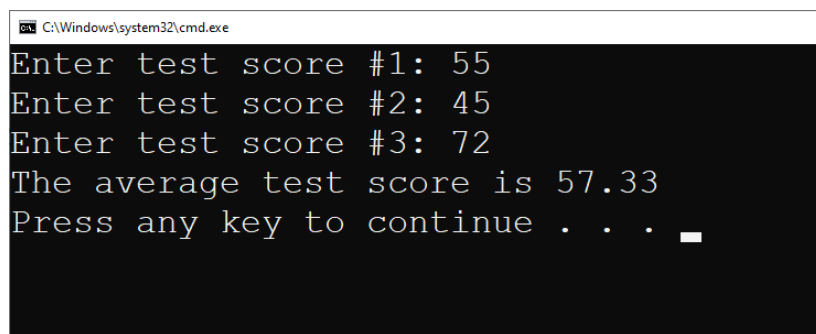
Write a class named *RetailItem* that holds data about an item in a retail store. The class should have the following fields:

Exercise 10

Design a **TestScores** class that has fields to hold three test scores. The class should have accessor and mutator methods for the test score fields, and a method that returns the average of the test scores.

Demonstrate the class by writing a separate program that creates an instance of the class. The program should ask the user to enter three test scores, which are stored in the TestScores object. Then the program should display the average of the scores, as reported by the TestScores object.

Your output should be similar to as shown below:



```
C:\Windows\system32\cmd.exe
Enter test score #1: 55
Enter test score #2: 45
Enter test score #3: 72
The average test score is 57.33
Press any key to continue . . .
```

Exercise 11

Write a Temperature class that will hold a temperature in Fahrenheit, and provide methods to get the temperature in Fahrenheit, Celsius, and Kelvin. The class should have the following field:

- ftemp – A double that holds a Fahrenheit temperature.

The class should have the following methods:

- setFahrenheit – The setFahrenheit method accepts a Fahrenheit temperature (as a double) and stores it in the ftemp field.
- getFahrenheit – Returns the value of the ftemp field, as a Fahrenheit temperature (no conversion required).
- getCelsius – Returns the value of the ftemp field converted to Celsius.
- getKelvin – Returns the value of the ftemp field converted to Kelvin.

Use the following formula to convert the Fahrenheit temperature to Celsius:

$$\text{Celsius} = 0.55555 \times (\text{Fahrenheit} - 32)$$

Use the following formula to convert the Fahrenheit temperature to Kelvin:

$$\text{Kelvin} = (0.55555 \times (\text{Fahrenheit} - 32)) + 273$$

Demonstrate the Temperature class by writing a separate program that asks the user for a Fahrenheit temperature. The program should create an instance of the Temperature class, with the value entered by the user passed to the appropriate methods. The program should then call the object's methods to display the temperature in Celsius and Kelvin.

```
C:\Windows\system32\cmd.exe
Enter the Fahrenheit temperature: 85
Celsius: 29.44
Kelvin: 302.44
Press any key to continue . . .
```

Exercise 12

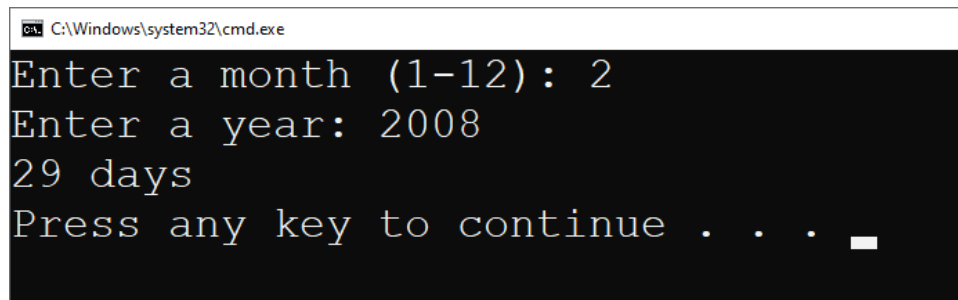
Write a class named `MonthDays`. The class's have the following fields:

- An integer for the month (1 5 January, 2 February, etc.).
- An integer for the year

The class should have appropriate method to set the information, and a method named `getNumberOfDays` that returns the number of days in the specified month. The method should use the following criteria to identify leap years:

1. Determine whether the year is divisible by 100. If it is, then it is a leap year if and if only it is divisible by 400. For example, 2000 is a leap year but 2100 is not.
2. If the year is not divisible by 100, then it is a leap year if and if only it is divisible by 4. For example, 2008 is a leap year but 2009 is not.

Demonstrate the class in a program that asks the user to enter the month (letting the user enter an integer in the range of 1 through 12) and the year. The program should then display the number of days in that month. See below for sample output from the program:



```
C:\Windows\system32\cmd.exe
Enter a month (1-12): 2
Enter a year: 2008
29 days
Press any key to continue . . .
```

Exercise 13

Write a Circle class that has the following fields:

- **radius**: a double
- **PI**: a final double initialized with the value 3.14159

The class should have the following methods:

- **setRadius**. A mutator method for the radius field.
- **getRadius**. An accessor method for the radius field.
- **getArea**. Returns the area of the circle, which is calculated as $\text{area} = \text{PI} * \text{radius} * \text{radius}$
- **getDiameter**. Returns the diameter of the circle, which is calculated as $\text{diameter} = \text{radius} * 2$
- **getCircumference**. Returns the circumference of the circle,
which is calculated as $\text{circumference} = 2 * \text{PI} * \text{radius}$

Write a program that demonstrates the Circle class by asking the user for the circle's radius, creating a Circle object, and then reporting the circle's area, diameter, and circumference.

C:\Windows\system32\cmd.exe

```
Enter the radius of a circle: 55
The circle's area is 9,503.31
The circle's diameter is 110.00
The circle's circumference is 345.57
Press any key to continue . . .
```


Exercise 14

Design a *SavingsAccount* class that stores a savings account's annual interest rate and balance. The class have methods for setting the balance and interest rate, subtracting the amount of a withdrawal, adding the amount of a deposit, and adding the amount of monthly interest to the balance. The monthly interest rate is the annual interest rate divided by twelve. To add the monthly interest to the balance, multiply the monthly interest rate by the balance, and add the result to the balance.

Test the class in a program that calculates the balance of a savings account at the end of a period of time. It should ask the user for the annual interest rate, the starting balance, and the number of months that have passed since the account was established. A loop should then iterate once for every month, performing the following:

- Ask the user for the amount deposited into the account during the month. Use the class method to add this amount to the account balance.
- Ask the user for the amount withdrawn from the account during the month. Use the class method to subtract this amount from the account balance.
- Use the class method to calculate the monthly interest.

After the last iteration, the program should display the ending balance, the total amount of deposits, the total amount of withdrawals, and the total interest earned.

```
C:\Windows\system32\cmd.exe
Enter the savings account's starting balance: 100
Enter the savings account's annual interest rate: 5
How many months have passed since the account was opened? 2
Enter the amount deposited during month 1: 100
Enter the amount withdrawn during month 1: 33
Enter the amount deposited during month 2: 100
Enter the amount withdrawn during month 2: 20
Total deposited: eur 200.00
Total withdrawn: eur 53.00
Interest earned:  eur 201.49
Ending balance: eur 448.49
Press any key to continue . . .
```