CpE-201

Object-Oriented Paradigm

Laboratory Manual

Prepared by:

Eng. Abeer Al-Fadhel

Eng. Aisha Al-Noori

Eng. Anwar Al-Ebrahim

Table of Contents

Lab 1: Introduction to Object-Oriented Paradigm Lab

Objectives:

1. Use input statements.
2. Use output statements (methods print, println and printf).
3. Use arithmetic, relational and equality operators.
4. Write decision-making statements.

Lab Instructions

Recognize and use conventional and standardized programming style and techniques for writing clear and readable programs.

Common programming errors:

- Forgetting one of the delimiters of a multiple-line comment.
- Not using the proper uppercase and lowercase letters for an identifier (Java is case sensitive).
- A public class must be placed in a file that has the same name as the class (in terms of both spelling and capitalization) plus the .java extension.
- If braces do not occur in matching pairs, the compiler indicates an error.
- Omitting the semicolon at the end of a statement.
- Splitting a statement in the middle of an identifier or a string.
- All import declarations must appear before the first class declaration in the file. Placing import declarations inside a class declaration's body or after a class declaration is a syntax error.
- Forgetting the left and/or right parenthesis for the condition in an if statement is a syntax error.
- Confusing the equality operator, ==, with the assignment operator, =, can cause a logic error or a syntax error.
- Placing a semicolon immediately after the right parenthesis of the condition in an if statement is normally a logic error.

Good Programming Practices:

- Begin your program with a comment that explains the purpose of program, author and date.
- Use blank lines and space characters to enhance program readability.
- By convention, always begin a class name with a capital letter and start each subsequent word in the class name with a capital letter.
- Use indentation (Tab key), to make your program more readable.
- Place a space after each comma (,) in an argument list to make programs more readable.
- Declare each variable on a separate line and insert a descriptive comment next to the declaration.
- Choose meaningful variable names to make your program to be self-documenting.
- By convention, variable name identifiers begin with a lowercase letter, and every word in the name after the first word begins with a capital letter.
- Place spaces on either side of a binary operator to make it stand out and make the program more readable.

Lab Exercise:

The body mass index (BMI) is often used to determine whether a person is overweight or underweight for his height. The body mass index (BMI) is calculated using the following formula:

$$BMI = \frac{weight}{height^2}$$

Where weight is measured in kilograms (kg) and height is measured in meters (m).

The following table shows different BMI categories:

BMI range	**Category**
Below 18.5	Underweight
From 18.5 to 24.9	Normal
From 25 to 30	Overweight
Over 30	Obese

Write a Java application based on the given flowchart in Figure 1. The application prompts the user to enter the weight and the height of a person, and then it calculates and displays the body mass index (BMI). In addition, the program should display a message indicating whether the person is underweight, normal, overweight or obese. You should make sure that the weight and the height are positive nonzero values, otherwise, output an error message. All values should be double.

<u>**Sample Output #1**</u>

```
Enter the height (in meter):  0
Enter the weight (in kilogram):  50
Invalid input! Height and weight should be nonzero positive values
```

<u>**Sample Output #2**</u>

```
Enter the height (in meter):  1.85
Enter the weight (in kilogram):  120
The BMI is 35.06
Obese
```

<u>**Sample Output #3**</u>

```
Enter the height (in meter):  1.65
Enter the weight (in kilogram):  65
The BMI is 23.88
Normal
```

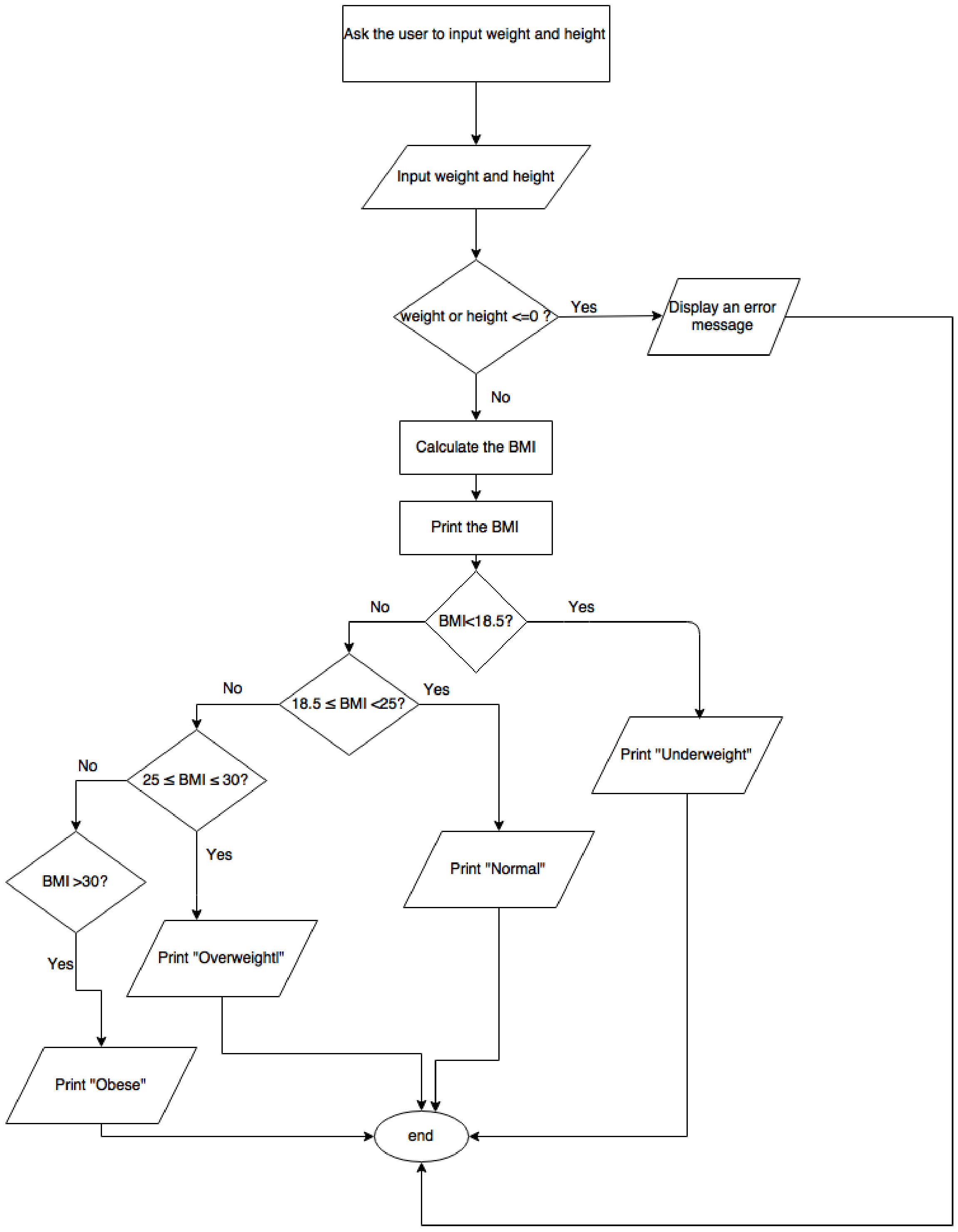

Ask the user to input weight and height
Input weight and height
weight or height <=0 ?
Yes
Display an error message
No
Calculate the BMI
Print the BMI
BMI<18.5?
No
Yes
18.5 ≤ BMI <25?
No
Yes
Print "Underweight"
25 ≤ BMI ≤ 30?
No
Yes
Print "Normal"
BMI >30?
Print "Overweightl"
Yes
Print "Obese"
end

Figure. 1

Lab 2: Introduction to classes and objects

Objectives:

1. Declare a class and use it to create an object.
2. Declare methods in a class to implement the class’s behaviors.
3. Declare instance variables in a class to implement the class’s attributes.
4. Call an object’s method to make that method perform its task.
5. Understand if, if…else and while statements.

Common programming errors:

- An attempt by a method that is not a member of a class to access a private member of that class is a compilation error.
- A compilation error occurs if a program attempts to initialize an object of a class by passing the wrong number or types of arguments to the class’s constructor.

Good Programming Practices:

- We prefer to list a class’s fields first, so that, as you read the code, you see the names and types of the variables before they’re used in the class’s methods.
- Place a blank line between method declarations to separate the methods and enhance program readability.

Lab Exercise:

Create a class called Course as described in Figure 3. The class includes three instance variables: courseID (type int), courseName (type String) and noOfStudents (type int). Your class should have the following:

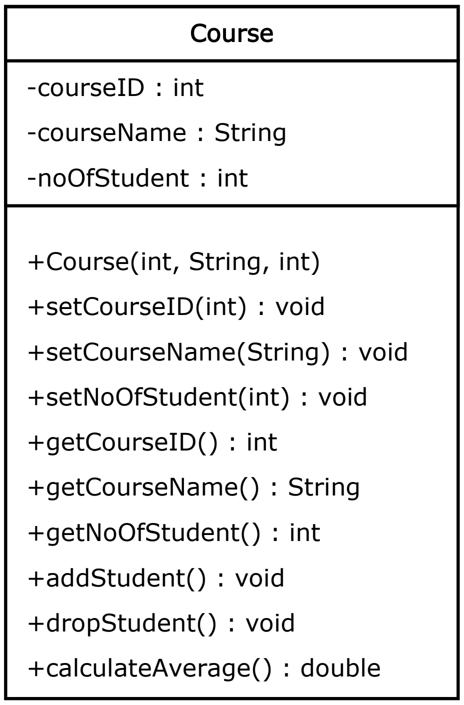


Figure. 3

- A constructor that initializes the three instance variables.
- set and get methods for each instance variable.
- A method addStudent to add one student to the course. The noOfStudent should be incremented by 1.
- A method dropStudent to drop one student from the course. The noOfStudent should be decremented by 1.
- A method calculateAverage to calculate the average mark of all students. In the method, you should ask the user to input the grades of the students. Grade is of type double.

Write a test application named CourseTest that demonstrates class Course capabilities. In the main method, do the following:

- Input the course ID, course name and number of students from the user.
- Create one Course object. Initialize the object using the data input by the user.
- Add a student to the course and display the number of students.

- Drop a student from the course and display the number of students.
- Calculate the average of the students in the class.

Sample Output

```
Enter the course ID: 201

Enter the course name: Object-Oriented Paradigm

Enter number of students: 5

One student has been added to the course

The number of students is 6

One student has been dropped from the course

The number of students is 5

Enter student 1 grade:  8.5
Enter student 2 grade:  6
Enter student 3 grade:  10
Enter student 4 grade:  5
Enter student 5 grade:  7
The average is 7.30
```

Lab 3: Switch Control Statement and random numbers

Objectives:

1. Use switch multiple-selection statement.
2. Use random-number generation.

Common programming errors:

- Forgetting a *break* statement when one is needed in a switch is a logic error.

Good Programming Practices:

- Although each *case* and the *default* case in a switch can occur in any order, place the *default* case last. When the *default* case is listed last, the break for that *case* is not required.

Lab Exercise:

Write a Java application based on the given flowchart in Figure 2. The application generates two integer numbers. The first one is in the range from 1 to 10 and the second one is selected randomly from the set <0, 2, 4, 6, 8>. Then, it displays a menu of the following choices:

1. Addition
2. Subtraction
3. Multiplication
4. Division (the second integer should not equal to zero). Assume integer division.
5. Compare the two numbers (determine if the first integer is larger, smaller or equal to the second integer)

The application should read the user choice and display the output.

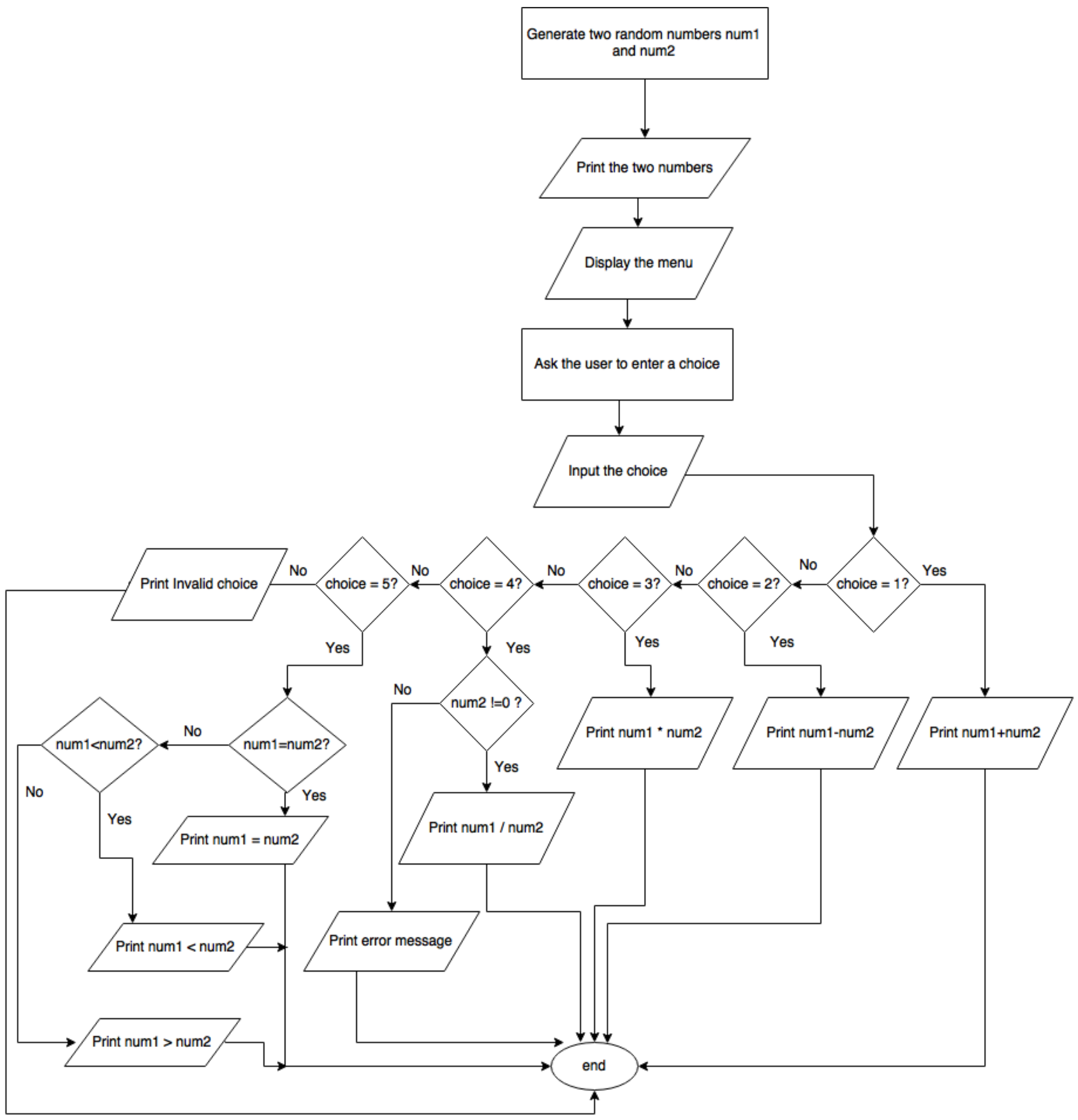


Figure. 2

Sample Output #1

```
The two numbers that are generated randomly are 9 and 4

Choose from the menu:
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Comparing two integers

Enter your choice: 1
9 + 4 = 13
```

Sample Output #2

```
The two numbers that are generated randomly are 7 and 0

Choose from the menu:
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Comparing two integers

Enter your choice: 4
ERROR: Cannot divide by zero!
```

Sample Output #3

```
The two numbers that are generated randomly are 3 and 6

Choose from the menu:
1- Addition
2- Subtraction
3- Multiplication
4- Division
5- Comparing two integers

Enter your choice: 5
3 < 6
```

Lab 4: Different types of repetitions structures

Objectives:

1. Use the *for* and *while* repetition statements.
2. Understand the difference between counter-controlled repetition and sentinel-controlled repetition.

Common programming errors:

- Using a semicolon after the while or for brackets will result in a logical error.
- Not providing, in the body of a *while* statement, an action that eventually causes the condition in the *while* to become false normally results in a logic error called an infinite loop (the loop never terminates).
- Using the value of a local variable before it's initialized results in a compilation error. All local variables must be initialized before their values are used in expressions.
- Using a sentinel value that is also a legitimate data value is a logic error.
- Omitting the braces that delimit a block can lead to logic errors, such as infinite loops.
- When a *for* statement's control variable is declared in the initialization section of the *for*'s header, using the control variable after the *for*'s body is a compilation error.

Good Programming Practices:

- Always using braces in an *if-else*, *while* or *for* helps prevent the accidental omission.
- In a sentinel-controlled loop, the prompts requesting data should remind the user of the sentinel.
- Place blank lines above and below repetition and selection control statements, and indent the statement bodies to enhance readability.

Lab Exercise 1:

Develop a Java application that allows the customer to buy online from an electronics store. The program starts by displaying the following message:

<<Welcome to BestBuy Electronics Store>>
Buy more than 500 KD and get a promotion

The program asks the customers to enter the number of items they want to buy, and then it displays the following menu that shows the available items in the store and the price of each item:

1. Desktop Computer	120 KD
2. Laptop	570 KD
3. Printer	55.900 KD
4. Scanner	60.500 KD

The customer should enter the item number as shown in sample output #1. At the end, the program displays the total price of all purchased items. If the total price exceeds 500 KD, the program chooses a promotion randomly for the customer. There are three kinds of promotions the store offers to the customers:

1. Free Mobile 2. Free Tablet 3. Free Camera

Lab Exercise 2:

Update the application that you have developed in lab exercise 1 by using the sentinel-controlled loop. The program will prompt the customer to enter the item number or -1 to stop the program as shown in sample output #2.

Sample Output #1

```
<< Welcome to BestBuy Electronics Store >>
Buy more than 500 KD and get a promotion
How many items do you want to buy? 3
Enter the item number:
1. Desktop Computer   120    KD
2. Laptop             570    KD
3. Printer            55.900 KD
4. Scanner            60.500 KD

Item 1:  3
you bought a printer

Item 2:  5
Invalid choice. Please enter a valid item number

Item 2:  1
you bought a desktop computer

Item 3:  2
you bought a laptop

The total price is 745.900
The total price exceeds 500 KD
You win a free Mobile!
```

Sample Output #2

```
<< Welcome to BestBuy Electronics Store >>
Buy more than 500 KD and get a promotion
Enter the item number (or -1 to stop):
1. Desktop Computer   120    KD
2. Laptop             570    KD
3. Printer            55.900 KD
4. Scanner            60.500 KD

Item 1:  3
you bought a printer

Item 2:  5
Invalid choice. Please enter a valid item number

Item 2:  1
you bought a desktop computer

Item 3:  2
you bought a laptop

Item 4:  -1

The total price is 745.900
The total price exceeds 500 KD
You win a free Mobile!
```

Lab 5: Methods and method calls

Objectives:

1. Understand methods and method calls.

Common programming errors:

- Declaring a method outside the body of a class declaration or inside the body of another method is a syntax error. Braces should mark the beginning and ending of a method.
- Placing a semicolon after the right parenthesis enclosing the parameter list of a method declaration is a syntax error.
- Re declaring a parameter as a local variable in the method's body is a compilation error.
- Forgetting to return a value from a method that should return a value and returning a value that does not match the method return type are compilation errors.
- A compilation error occurs when a local variable is declared more than once in a method.
- A compilation error occurs if the number and order of method parameters in the method call does not match the method header.

Good Programming Practices:

- Declare variables as close to where they're first used as possible.

Lab Exercise:

Write a program that implements the following options on a given 5-digits number based on the user choice:

1- Palindrome Check
2- Reverse Digit
3- Prime Check
4- Palindromic Prime
5- Sum of even digits
6- Exit

Implement the method as follows:

1. Palindrome Check

   This method (palindrome) should return true if the given number is a palindrome and false otherwise. A number is considered a palindrome if it's reverse is the same as itself. Use method reverse to implement the palindrome method

2. Reverse Digit

   This method (reverseDigit) will take a number and reverse its digits.

3. Prime Check

   This method (prime) should return true if the given number is prime number and false otherwise. A number is considered prime if it cannot be divided by any number before it.

4. Palindromic Prime

   This method (palindromicPrime) should return true if the given number is a palindrome and prime number and false otherwise. Use method prime and palindrome to check the number.

5. Sum of even digits

   This method (sumEven) should return the sum of even digit of the given number.

6. Exit

   This option will exit the program

Sample Output

```
Please enter a number: 12321

1. Palindrome Check

2. Reverse Digit

3. Prime Check

4. Palindromic Prime
```

```
5. Sum of even digits

6. Exit

Choose an option: 1

The number is a palindrome

Please enter a number: 12321

1. Palindrome Check

2. Reverse Digit

3. Prime Check

4. Palindromic Prime

5. Sum of even digits

6. Exit

Choose an option: 2

The number is 12321

Please enter a number: 12321

1. Palindrome Check

2. Reverse Digit

3. Prime Check

4. Palindromic Prime

5. Sum of even digits

6. Exit

Choose an option: 3

The number is not a prime

Please enter a number: 12321

1. Palindrome Check

2. Reverse Digit

3. Prime Check

4. Palindromic Prime

5. Sum of even digits

6. Exit

Choose an option: 4

The number is not palindromic prime
```

```
Please enter a number: 12321
1. Palindrome Check
2. Reverse Digit
3. Prime Check
4. Palindromic Prime
5. Sum of even digits
6. Exit
Choose an option: 5
The sum of even digits is 4
Please enter a number: 12321
1. Palindrome Check
2. Reverse Digit
3. Prime Check
4. Palindromic Prime
5. Sum of even digits
6. Exit
Choose an option: 6
Thank you!
```

Lab 6: Arrays

Objectives:

1. Declare and initialize arrays.
2. Pass arrays to methods and return arrays.

Common programming errors:

- In an array declaration, specifying the number of elements in the square brackets of the declaration (e.g. int c[12];) is a syntax error.

Good Programming Practices:

- Constant variables (e.g. ARRAY_LENGTH) make programs more readable than programs that use literal values (e.g. 10).

Lab Exercise:

Write a Java application called **GradeStatistics**, which keeps the grades of 20 students and displays the average, minimum, maximum, standard deviation and the histogram of the grades. The program should keep the grades in an integer array. Your program should include the following methods:

1. initArray
   takes an integer array as a parameter and initializes it with random numbers in the range 0 – 99.
2. Maximum
   takes an integer array as a parameter and returns the maximum grade.
3. Minimum
   takes an integer array as a parameter and returns the minimum grade.
4. Average
   takes an integer array as a parameter and returns the average of the grades as double.

5. standardDev

   takes an integer array as a parameter and returns the standard deviation of the grades as double.

$$deviation = \sqrt{\frac{\sum_{i=1}^{n}(x_i - mean)^2}{n-1}}$$

6. printArray

   takes an integer array and prints the array elements.
7. computeHistogram

   takes an integer array as parameter and returns a new integer array. The method counts the frequency of each grade and stores the count value in the new array. Assume the first element of the new array stores the frequency of grades in the range 0 to 9, the second element stores the frequency of grades in the range 10 to 19, etc.
8. printHistogram

   takes an integer array as parameter and prints the histogram horizontally. Hint: you can call method computeHistogram.

Write a main method to print the results (use method calls).

Sample Output

```
The grades are:
46 4 91 26 68 91 63 26 31 17 10 33 50 45 12 24 24 80 59 12

The maximum grade is  91
The minimum grade is  4
The average is  40.60
The standard deviation is 27.10

The grades histogram:
 0- 9:   *
10-19:   ****
20-29:   ****
30-39:   **
40-49:   **
50-59:   **
60-69:   **
70-79:
80-89:   *
90-99:   **
```

Lab 7: Classes and Objects – A Deeper Look

Objectives:

1. Use static variables and methods.
2. Create an array of reference-type elements (each element of the array is a reference to an object).
3. Understand the composition in which a class has references to objects of other classes as members.

Lab Exercise:

Implement the following classes:

1) Class **Point that has the following properties:**
    - Two instance variables x and y (of type integer) that represent the x-y coordinate of a point.
    - A default constructor that initializes the instance variables to the default values.
    - A constructor that initializes the instance variables to the values specified as parameters.
    - Set and get methods for the instance variables.
    - A method distance that takes a reference to a Point object myPoint as a parameter and returns the distance between this Point to the given instance myPoint passed as a parameter.
    - A method distance that takes two integer parameters and returns the distance between this Point to the given point with coordinates specified in the parameters.
      Note that the distance between the two points (x_1, y_1) and (x_2, y_2) is given as $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.
    - A method toString that returns the string representation of a Point given as point (x, y).

2) Class **Line that has the following properties:**

- Two instance variables start and end that are references to Point objects. The two points represent the end points of a Line segment.
- A class (static) variable counter (of type integer) that specifies the number of Line objects created.
- A constructor that initializes the instance variables start and end to the references specified as parameters and increments the static variable counter by one to indicate that a Line has been created.
- A static method getCounter that returns the static variable counter.
- A method slope that returns the slope of the Line.
  Note that the slope of the Line with the two end points (x_1, y_1) and (x_2, y_2) is given as $\frac{y_1-y_2}{x_1-x_2}$.
- A method isParallel that takes a reference to a Line object myLine as a parameter and returns true if the two lines are parallel or false otherwise. The two lines are parallel if they have the same slope.
- A method toString that returns the string representation of a Line given as Line from (x_1, y_1) to (x_2, y_2).

2) Class **LineTest** that has a main method to test your classes. The main method should do the following:

- Create an array of three references to Line objects.
- Input the two end points of each Line, create the objects and store them in the array.
- Print the number of lines created.
- Use a loop to print the string representation of each line and its slope.
- Use a loop to check whether the first line in the array is parallel to any other line and show the results.
- Create two Point objects and test the two versions of the distance method.

Sample Output

```
Line 1
Enter x-y coordinate of the start point: 1 2
Enter x-y coordinate of the end point: 3 4

Line 2
Enter x-y coordinate of the start point: 4 8
Enter x-y coordinate of the end point: 2 6

Line 3
Enter x-y coordinate of the start point: 2 1
Enter x-y coordinate of the end point: 5 6

The number of lines created is 3

Line from point (1, 2) to point (3, 4)
the slope is 1.00

Line from point (4, 8) to point (2, 6)
the slope is 1.00

Line from point (2, 1) to point (5, 6)
the slope is 1.67

Line from point (1, 2) to point (3, 4) is parallel to Line from point
(4, 8) to point (2, 6)

The distance between point (1, 2) and point (2, 4) is 2.24

The distance between point (1, 2) and point (2, 4) is 2.24
```

Lab 8: Characters and String manipulation methods

Objectives:

Test the methods of class String and class Character

Lab Exercise:

Write an application that reads two text values from the user and do some string manipulation. The strings should be stored in objects of type String s1 and s2. You need to show the following menu for the user:

1. Reverse
2. Remove a character
3. Number of uppercase letters
4. Number of digits
5. Compare two string
6. Exit

Each of the above options should be done using the following methods:

- **Reverse**

  Takes a String and reverse it. The method returns the reversed String.

- **Remove a character**

  This method takes a character and a String, and removes all occurrences of that character in the String. The method returns the new String. Java is case sensitive so removing 'a' is not like removing 'A'. You need to handle the case in which the character is not found.

- **Number of uppercase letters**

  This method takes a String and returns the number of uppercase letters in the String.

- **Number of digits**

  This method takes a String and returns the number of uppercase letters in the String.

- Compare two string

  This method checks whether the two strings passed are equal or not. Use String comparisons methods (s1.equals(s1) or s1.compareTo(s2)). The method returns true if the two Strings are equal and false otherwise.

<u>Sample Output</u>

```
Enter a string: Database1

Enter a string: Digital

1.     Reverse

2.     Remove a character

3.     Number of uppercase letters

4.     Number of digits

5.     Compare two string

6.     Exit

Choose an option:1

1esabataD


1.     Reverse

2.     Remove a character

3.     Number of uppercase letters

4.     Number of digits

5.     Compare two string

6.     Exit

Choose an option:2

What is the letter to be removed? a

1esbtD


1.     Reverse

2.     Remove a character

3.     Number of uppercase letters

4.     Number of digits

5.     Compare two string

6.     Exit
```

```
Choose an option:3
The number of upper case letters is 1


1.    Reverse
2.    Remove a character
3.    Number of uppercase letters
4.    Number of digits
5.    Compare two string
6.    Exit
Choose an option:4


The number of digits is 1
1.    Reverse
2.    Remove a character
3.    Number of uppercase letters
4.    Number of digits
5.    Compare two string
6.    Exit
Choose an option:5
1esbtD not equal to Digital
1.    Reverse
2.    Remove a character
3.    Number of uppercase letters
4.    Number of digits
5.    Compare two string
6.    Exit
Choose an option:6
Thank you!
```

Lab 9: Inheritance and Polymorphism

Objectives:

1. Understand the concept of inheritance.
2. Understand the notion of superclasses and subclasses
3. Understand the concept of polymorphism.
4. Distinguish between abstract and concrete classes.
5. Declare abstract methods to create abstract classes.

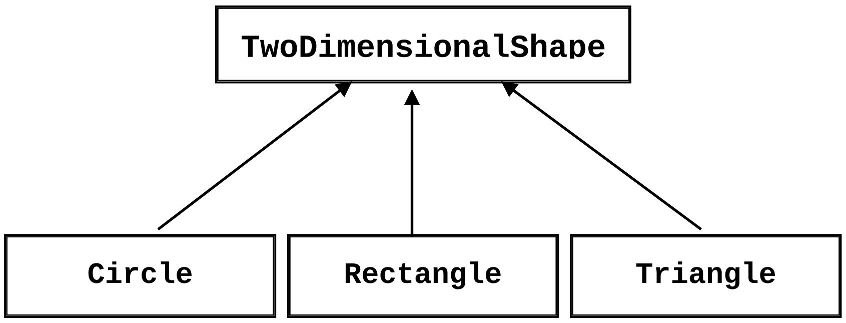
Common Programming Errors:

- Attempting to create an object of an abstract class is a compilation error.
- Failure to implement a superclass's abstract methods in a subclass is a compilation error unless the subclass is also declared abstract.
- Assigning a superclass variable to a subclass variable (without an explicit cast) is a compilation error.

Lab Exercise:

1) Inheritance:

Implement the following classes' hierarchy:



The hierarchy has the following classes:

- Class **TwoDimensionalShape** is the superclass of the hierarchy. It has one instance variable shapeName (of type String). It has a constructor that initializes shapeName to the value specified in the parameter. It has one method toString that returns the string representation of the shape name.

- Class **Circle** is derived from TwoDimensionalShape. It has one instance variable radius (of type double). It has a constructor that calls the super class constructor to initialize the shape name and it initializes the radius to the value specified in the parameter. It has two methods: a method getArea that returns the area of the circle and a method toString that overrides the superclass version and returns a string representation of the shape name and the area.
- Class **Rectangle** is derived from TwoDimensionalShape. It has two instance variables length and width (both of type double). It has a constructor that calls the super class constructor to initialize the shape name and it initializes the length and width to the values specified in the parameters. It has two methods: a method getArea that returns the area of the rectangle and a method toString that overrides the superclass version and returns a string representation of the shape name and the area.
- Class **Triangle** is derived from TwoDimensionalShape. It has two instance variables base and height (both of type double). It has a constructor that calls the super class constructor to initialize the shape name and it initializes the base and height to the values specified in the parameters. It has two methods: a method getArea that returns the area of the triangle and a method toString that overrides the superclass version and returns a string representation of the shape name and the area.
- Write a test class named **TwoDimensionalShapeTest** that has a main method to test your classes. In the main method, prompt the user to input a two-dimensional shape name, the radius of a circle, the length and width of a rectangle and the base and height of a triangle. Create TwoDimensionalShape, Circle, Rectangle and Triangle objects and initialize the objects with the values entered by the user. Call toString method for each object and print the output.

Area formula:

Circle Area = $\pi * radius^2$

Rectangle Area = $length * width$

Triangle Area = $\frac{1}{2} * base * height$

<u>**Sample Output**</u>

```
Enter the two dimensional shape name: 2D shape
Enter the radius of the circle: 2
Enter the length of the rectangle: 2
Enter the width of the rectangle: 4
Enter the base of the triangle: 4
Enter the height of the triangle: 8

This is a 2D shape
This is a Circle. Its area is 12.57 squared cm
This is a Rectangle. Its area is 8.00 squared cm
This is a Triangle. Its area is 16.00 squared cm
```

2) Polymorphism:

Modify the previous exercise to test the polymorphism as follows:

- An abstract class **TwoDimensionalShape** is the superclass of the hierarchy. It has one instance variable shapeName (of type String). It has a constructor that initializes shapeName to the value specified in the parameter. It has two methods toString that returns the string representation of the shape name and an abstract method getArea.
- Class **Circle** is derived from TwoDimensionalShape. It has one instance variable radius (of type double). It has a constructor that calls the super class constructor to initialize the shape name and it initializes the radius to the value specified in the parameter. It has two methods: a method getArea that implements the abstract method

and returns the area of the circle and a method toString that overrides the superclass version and returns a string representation of the shape name and the area.

- Class **Rectangle** is derived from TwoDimensionalShape. It has two instance variables length and width (both of type double). It has a constructor that calls the super class constructor to initialize the shape name and it initializes the length and width to the values specified in the parameters. It has two methods: a method getArea that implements the abstract method and returns the area of the rectangle and a method toString that overrides the superclass version and returns a string representation of the shape name and the area.
- Class **Triangle** is derived from TwoDimensionalShape. It has two instance variables base and height (both of type double). It has a constructor that calls the super class constructor to initialize the shape name and it initializes the base and height to the values specified in the parameters. It has two methods: a method getArea that implements the abstract method and returns the area of the triangle and a method toString that overrides the superclass version and returns a string representation of the shape name and the area.
- Write a test class named **TwoDimensionalShapeTest** that has a main method to test your classes. You will use superclass references to manipulate the subclass objects polymorphically. In the main method, create an array of TwoDimensionalShape references, the array holds an instance of a Circle, an instance of Rectangle and an instance of Triangle objects. Prompt the user to input the radius of a circle, the length and width of a rectangle and the base and height of a triangle. Create Circle, Rectangle and Triangle objects and initialize them with the values entered by the user. Use a loop that goes through the TwoDimensionalShape array, call toString method for each object and print the output.

Sample Output

```
Enter the radius of the circle: 2
Enter the length of the rectangle: 2
Enter the width of the rectangle: 4
Enter the base of the triangle: 4
Enter the height of the triangle: 8
This is a Circle. Its area is 12.57 squared cm
This is a Rectangle. Its area is 8.00 squared cm
This is a Triangle. Its area is 16.00 squared cm
```

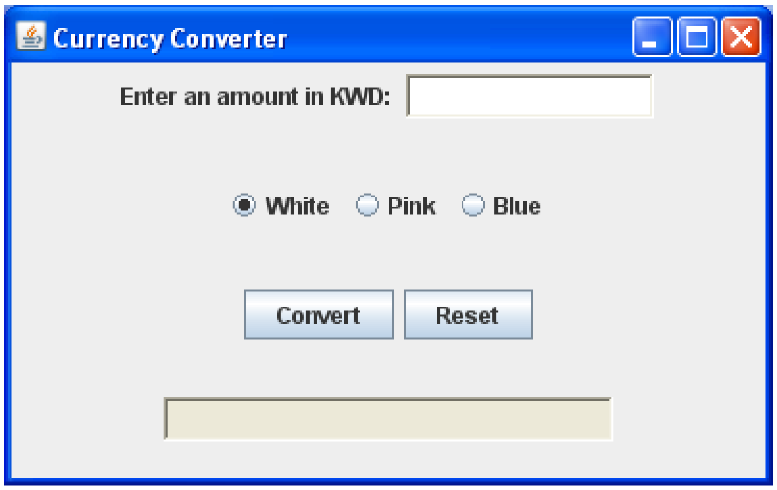
Lab 10: Graphical User Interface (Frames only)

Objectives:

1. Use GUI libraries to build a user friendly interface.
2. Create and manipulate labels, text fields, buttons and panels.
3. Use layout managers to arrange GUI components

Lab Exercise:

Write a GUI-based Java application that implements the "Currency Converter" shown below. In this exercise you don't have to provide any functionality.



You will code the following items:

1. A window titled "Currency Converter".
2. A Label and two text fields.
3. Three Radio Buttons: **White**, **Pink** and **Blue**. The White Radio Button will be selected initially
4. Two Buttons: **Convert** and **Reset**.
5. Panels to arrange the components.
6. Set the frame size to 400 by 250.

Additional Information:

1. To import all classes in a package, use * as follows:
    - import java.awt.*;
    - import java.awt.event.*;
    - import javax.swing.*;

2. To set the frame Background to Color white, use
   getContentPane().setBackground(Color.white)

3. To set a Panel "P" Background to Color white, use
   P.setBackground(Color.white);

Lab 11: Events generated by GUIs

Objectives:

Handling events generated by user interactions with the GUIs

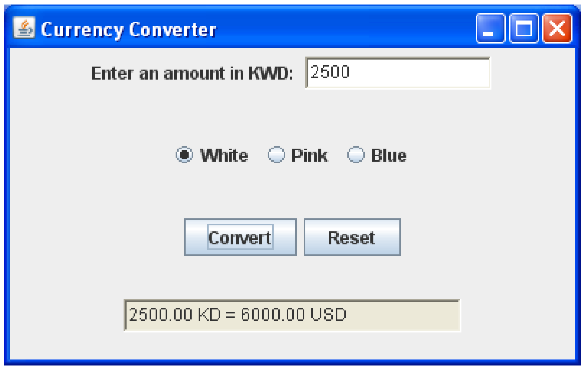
Lab Exercise:

Modify your program of Lab 10 by adding inner classes that handle the following events:

1. When the user enters an amount (in Kuwaiti Dinars) in the text filed and clicks on the **Convert** button, the second text filed shows the amount in United States Dollars. (1 KWD = 2.4 USD).
2. The **Reset** Button should clear the Text Fields.
3. The user should be able to change the background color using the Radio buttons.

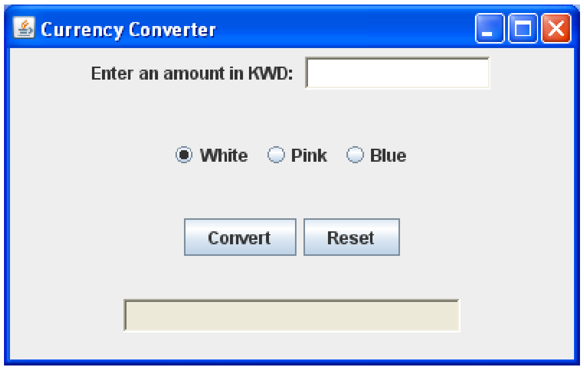
<u>Sample Output</u>

Enter an amount and click on **Convert**:

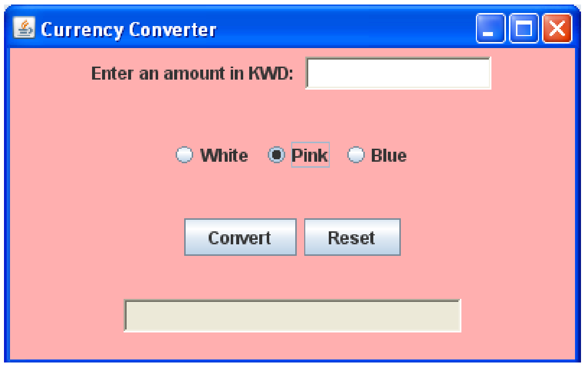


```
2500
2500.00 KD = 6000.00 USD
```

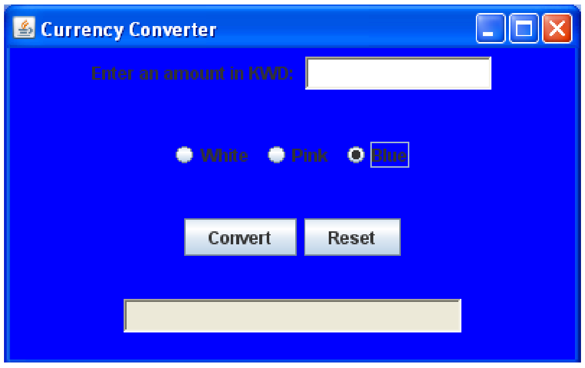
Click on **Reset**:



Click on **Pink**:



Click on **Blue**:



Reference

- Java How To Program, Deitel and Deitel, eighth edition.