PATUAKHALI SCIENCE & TECHNOLOGY UNIVERSITY

SUBMITTED TO:

Prof. Dr. Md Samsuzzaman

Department of Computer and Communication Engineering

Faculty of Computer Science and Engineering

SUBMITTED BY:

NAME : Sakib Hasan

ID:2102052

Faculty of Computer Science and Engineering

Date of submission: **11-11-2023**

Assignment: Assignment based on chapter (6)

Self-Review Exercises

6.1 Fill in the blanks in each of the following statements:

Fill in the blanks in each of the following statements:

a) A method is invoked with a(n) . method call.

b) A variable known only within the method in which it’s declared is called a(n) . local variable.

c) The return statement in a called method can be used to pass the value of an expression back to the calling method.

d) The keyword void indicates that a method does not return a value.

e) Data can be added or removed only from the top of a stack.

f) Stacks are known as Last in First out (LIFO) data structures; the last item pushed (inserted) onto the stack is the first item popped (removed) from the stack.

g) The three ways to return control from a called method to a caller are return; or return expression; or encountering the closing right brace of a method.

h) An SecureRandom object of class produces truly random numbers.

i) The method-call stack contains the memory for local variables on each invocation of a method during a program’s execution. This data, stored as a portion of the method-call stack, is known as the Stack frame or activation record of the method call.

j) If there are more method calls than can be stored on the method-call stack, an error known as a(n) Stack Overflow occurs.

k) The Scope of a declaration is the portion of a program that can refer to the entity in the declaration by name.

l) It’s possible to have several methods with the same name that each operate on different types or numbers of arguments. This feature is called method overloading .

6.2

For the class Craps in Fig. 6.8, state the scope of each of the following entities:

a) the variable randomNumbers.

b) the variable die1.

c) the method rollDice.

d) the method main.

e) the variable sumOfDice.

Answers:

a) the variable `randomNumbers`: The scope of `randomNumbers` is the entire class `Craps` because it is declared as a private static final field at the class level. It is accessible from any method within the `Craps` class.

b) the variable `die1`: The scope of `die1` is the method `rollDice`. It is declared and used within the `rollDice` method and is not accessible outside of that method.

c) the method `rollDice`: The scope of `rollDice` is the entire class `Craps` as it is declared as a method of the class. It can be called from any other method within the class, including the `main` method.

d) the method `main`: The scope of `main` is the entire class `Craps` as it is the main method and an entry point for the program. It can access the class fields and call other methods within the class.

e) the variable `sumOfDice`: The scope of `sumOfDice` is the main method (`main`). It is declared within the `main` method and is not accessible outside of it.

6.3

Write an application that tests whether the examples of the Math class method calls shown in Fig. 6.2 actually produce the indicated results.

```
// Exercise 6.3: MathTest.java
// Testing the Math class methods.
public class MathTest
{
public static void main(String[] args)
{
```

```java
System.out.printf("Math.abs(23.7) = %f%n", Math.abs(23.7));
System.out.printf("Math.abs(0.0) = %f%n", Math.abs(0.0));
System.out.printf("Math.abs(-23.7) = %f%n", Math.abs(-23.7));
System.out.printf("Math.ceil(9.2) = %f%n", Math.ceil(9.2));
System.out.printf("Math.ceil(-9.8) = %f%n", Math.ceil(-9.8));
System.out.printf("Math.cos(0.0) = %f%n", Math.cos(0.0));
System.out.printf("Math.exp(1.0) = %f%n", Math.exp(1.0));
System.out.printf("Math.exp(2.0) = %f%n", Math.exp(2.0));
System.out.printf("Math.floor(9.2) = %f%n", Math.floor(9.2));
System.out.printf("Math.floor(-9.8) = %f%n", Math.floor(-9.8));
System.out.printf("Math.log(Math.E) = %f%n", Math.log(Math.E));
System.out.printf("Math.log(Math.E * Math.E) = %f%n",
Math.log(Math.E * Math.E));
System.out.printf("Math.max(2.3, 12.7) = %f%n", Math.max(2.3, 12.7));
System.out.printf("Math.max(-2.3, -12.7) = %f%n",
Math.max(-2.3, -12.7));
System.out.printf("Math.min(2.3, 12.7) = %f%n", Math.min(2.3, 12.7));
System.out.printf("Math.min(-2.3, -12.7) = %f%n",
Math.min(-2.3, -12.7));
System.out.printf("Math.pow(2.0, 7.0) = %f%n", Math.pow(2.0, 7.0));
System.out.printf("Math.pow(9.0, 0.5) = %f%n", Math.pow(9.0, 0.5));
System.out.printf("Math.sin(0.0) = %f%n", Math.sin(0.0));
System.out.printf("Math.sqrt(900.0) = %f%n", Math.sqrt(900.0));
System.out.printf("Math.tan(0.0) = %f%n", Math.tan(0.0));
} // end main } // end class MathTes
```

6.4 **Give the method header for each of the following methods:**

a) Method hypotenuse, which takes two double-precision, floating-point arguments side1 and side2 and returns a double-precision, floating-point result.

b) Method smallest, which takes three integers x, y and z and returns an integer.

c) Method instructions, which does not take any arguments and does not return a value. [Note: Such methods are commonly used to display instructions to a user.]

d) Method intToFloat, which takes integer argument number and returns a float.

Answer:

a) double hypotenuse(double side1, double side2)

b) int smallest(int x, int y, int z)

c) void instructions()

d) float intToFloat(int number)

6.5

Find the error in each of the following program segments. Explain how to correct the error.

a) void g() { System.out.println("Inside method g");

void h() { System.out.println("Inside method h");

 }

}

b) int sum(int x, int y) { int result; result = x + y; }

c) void f(float a); { float a; System.out.println(a); }

d) void product() { int a = 6, b = 5, c = 4, result; result = a * b * c;

System.out.printf("Result is %d%n", result);

 return result;

```
 }
```

Answer:

a) Error: Method h is declared within method g.

Correction: Move the declaration of h outside the declaration of g.

b) Error: The method is supposed to return an integer, but does not.

Correction: Delete the variable result, and place the statement return x + y; in the method, or add the following statement at the end of the method body: return result;

c) Error: The semicolon after the right parenthesis of the parameter list is incorrect, and the parameter a should not be redeclared in the method.

Correction: Delete the semicolon after the right parenthesis of the parameter list, and delete the declaration float a;.

d) Error: The method returns a value when it's not supposed to.

Correction: Change the return type from void to int.

6.6

6.6 Declare method sphereVolume to calculate and return the volume of the sphere. Use the following statement to calculate the volume:

double volume = (4.0 / 3.0) * Math.PI * Math.pow(radius, 3)

Write a Java application that prompts the user for the double radius of a sphere, calls sphereVolume to calculate the volume and displays the result.

Answer:

```
// Exercise 6.6: Sphere.java
// Calculate the volume of a sphere.
import java.util.Scanner;

```

```
public class Sphere
{
// obtain radius from user and display volume of sphere
public static void main(String[] args)
{
Scanner input = new Scanner(System.in);

System.out.print("Enter radius of sphere: ");
double radius = input.nextDouble();

System.out.printf("Volume is %f%n", sphereVolume(radius));
} // end method determineSphereVolume

// calculate and return sphere volume
public static double sphereVolume(double radius)
{
double volume = (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
return volume;
} // end method sphereVolume
} // end class Sphere
```

6.7

What is the value of x after each of the following statements is executed?

a) x = Math.abs(-7.5);

b) x = Math.floor(5 + 2.5);

c) x = Math.abs(9) + Math.ceil(2.2);

d) x = Math.ceil(-5.2);

e) x = Math.abs(-5) + Math.abs(4);

f) x = Math.ceil(-6.4) - Math.floor(5.2);

g) x = Math.ceil(-Math.abs(-3 + Math.floor(-2.5)));

Answer:

a) `x = Math.abs(-7.5);`

Result: `x` is assigned the absolute value of -7.5, so `x` becomes 7.5.

b) `x = Math.floor(5 + 2.5);`

Result: `x` is assigned the floor value of the sum of 5 and 2.5, so `x` becomes 7.0.

c) `x = Math.abs(9) + Math.ceil(2.2);`

Result: `x` is assigned the sum of the absolute value of 9 and the ceiling value of 2.2, so `x` becomes 11.0.

d) `x = Math.ceil(-5.2);`

Result: `x` is assigned the ceiling value of -5.2, so `x` becomes -5.0.

e) `x = Math.abs(-5) + Math.abs(4);`

Result: `x` is assigned the sum of the absolute values of -5 and 4, so `x` becomes 9.

f) `x = Math.ceil(-6.4) - Math.floor(5.2);`

Result: `x` is assigned the difference between the ceiling value of -6.4 and the floor value of 5.2, so `x` becomes -6.0.

g) `x = Math.ceil(-Math.abs(-3 + Math.floor(-2.5)));`

Result: `x` is assigned the ceiling value of the negation of the absolute value of the sum of -3 and the floor value of -2.5, so `x` becomes -3.0.

In summary:

a) 7.5

b) 7.0

c) 11.0

d) -5.0

e) 9

f) -6.0

g) -3.0

6.8 **(Parking Charges) A parking garage charges a $2.00 minimum fee to park for up to three hours. The garage charges an additional $0.50 per hour for each hour or part thereof in excess of three hours. The maximum charge for any given 24-hour period is $10.00. Assume that no car parks for longer than 24 hours at a time. Write an application that calculates and displays the parking charges for each customer who parked in the garage yesterday. You should enter the hours parked for each customer. The program should display the charge for the current customer and should calculate and display the running total of yesterday's receipts. It should use the method calculateCharges to determine the charge for each customer.**

Answer:

```
import java.util.Scanner;
```

```
public class ParkingCharges {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        double totalReceipts = 0.0;

        System.out.print("Enter the number of customers: ");
        int numberOfCustomers = scanner.nextInt();

        for (int i = 1; i <= numberOfCustomers; i++) {
            System.out.printf("Enter hours parked for customer %d: ", i);
            int hoursParked = scanner.nextInt();

            double charge = calculateCharges(hoursParked);

            System.out.printf("Customer %d parking charge: $%.2f%n", i, charge);

            totalReceipts += charge;
        }

        System.out.printf("Total receipts for yesterday: $%.2f%n", totalReceipts);

        scanner.close();
    }
    private static double calculateCharges(int hours) {
        double minimumCharge = 2.00;
        double hourlyRate = 0.50;
        double maximumCharge = 10.00;

        // Calculate parking charge
        if (hours <= 3) {
            return minimumCharge;
        } else if (hours <= 24) {
            double additionalHours = hours - 3;
            double additionalCharge = Math.min(additionalHours * hourlyRate, maximumCharge -
minimumCharge);
            return minimumCharge + additionalCharge;
        } else {
            return maximumCharge; // Maximum charge for any 24-hour period
        }
    }
}
```

6.9 **(Rounding Numbers) Math.floor can be used to round values to the nearest integer—e.g., y = Math.floor(x + 0.5); will round the number x to the nearest integer and assign the result to y. Write an application that reads double values and uses the preceding statement to round each of the numbers to the nearest integer. For each number processed, display both the original number and the rounded number.**

Answer:

```
import java.util.Scanner;

public class RoundingNumbers {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a double value (or a non-double value to exit): ");

        while (scanner.hasNextDouble()) {
            double originalNumber = scanner.nextDouble();
            double roundedNumber = Math.floor(originalNumber + 0.5);

            System.out.printf("Original number: %.2f, Rounded number: %.0f%n", originalNumber,
roundedNumber);

            System.out.print("Enter another double value (or a non-double value to exit): ");
        }

        System.out.println("Program exiting. Goodbye!");
        scanner.close();
    }
}
```

6.10 (Rounding Numbers) To round numbers to specific decimal places, use a statement like y = Math.floor(x * 10 + 0.5) / 10; which rounds x to the tenths position (i.e., the first position to the right of the decimal point), or y = Math.floor(x * 100 + 0.5) / 100; which rounds x to the hundredths position (i.e., the second position to the right of the decimal point). Write an application that defines four methods for rounding a number x in various ways:

a) roundToInteger(number)

b) roundToTenths(number)

c) roundToHundredths(number)

d) roundToThousandths(number) For each value read,

your program should display the original value, the number rounded to the nearest integer, the number rounded to the nearest tenth, the number rounded to the nearest hundredth and the number rounded to the nearest thousandth.

Answer:

```
import java.util.Scanner;


public class RoundingNumbers {


  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);


    System.out.print("Enter a double value (or a non-double value to exit): ");


    while (scanner.hasNextDouble()) {
      double originalNumber = scanner.nextDouble();


      displayRoundedValues(originalNumber);


      System.out.print("Enter another double value (or a non-double value to exit): ");
    }
    System.out.println("Program exiting. Goodbye!");

    scanner.close();

}


private static void displayRoundedValues(double x) {

    System.out.printf("Original number: %.3f%n", x);

    System.out.printf("Rounded to nearest integer: %.0f%n", roundToInteger(x));

    System.out.printf("Rounded to nearest tenth: %.1f%n", roundToTenths(x));

    System.out.printf("Rounded to nearest hundredth: %.2f%n", roundToHundredths(x));

    System.out.printf("Rounded to nearest thousandth: %.3f%n", roundToThousandths(x));

    System.out.println();

}


private static double roundToInteger(double x) {

    return Math.floor(x + 0.5);

}


private static double roundToTenths(double x) {

    return Math.floor(x * 10 + 0.5) / 10;

}


private static double roundToHundredths(double x) {

    return Math.floor(x * 100 + 0.5) / 100;

}
```

```
    private static double roundToThousandths(double x) {
        return Math.floor(x * 1000 + 0.5) / 1000;
    }
}
```

6.11 **Answer each of the following questions:**

a) What does it mean to choose numbers “at random”?

b) Why is the nextInt method of class SecureRandom useful for simulating games of chance?

c) Why is it often necessary to scale or shift the values produced by a SecureRandom object?

d) Why is computerized simulation of real-world situations a useful technique?

Answer:

a) Choosing numbers "at random" means selecting values in a way that each possible outcome has an equal chance of occurring. In programming, this often involves using algorithms or functions that generate pseudo-random numbers, as true randomness can be difficult to achieve in a deterministic computer system.

b) The `nextInt` method of the `SecureRandom` class is useful for simulating games of chance because it provides a secure and unpredictable source of random integers. The `SecureRandom` class is designed to be cryptographically secure, meaning it is resistant to prediction and tampering. In games of chance, it's crucial to have unpredictable outcomes, and `SecureRandom` helps achieve this by providing a high-quality source of random numbers.

c) Scaling or shifting the values produced by a `SecureRandom` object may be necessary to map the generated random numbers to a specific range or set of values. For example, if you need random numbers between 1 and 100, you can generate a random number using `nextInt` and then scale and shift it accordingly. Scaling involves multiplying the random number by a scaling factor, and shifting involves adding an offset. This ensures that the random numbers meet the requirements of the specific application.

d) Computerized simulation of real-world situations is a useful technique because it allows researchers, scientists, and engineers to model and analyze complex systems without the need for real-world experimentation. Simulations can provide insights into the behavior of systems, help test hypotheses, and predict outcomes under different conditions. It is particularly valuable when real-world experiments are expensive, time-consuming, or impractical. Simulations allow for repeated testing, control over variables, and the ability to explore scenarios that may be difficult or dangerous to replicate in reality.

6.12 **Write statements that assign random integers to the variable n in the following ranges:**

a) 2 ≤ n ≤ 6.

b) 4 ≤ n ≤ 50.

c) 0 ≤ n ≤ 7.

d) 1000 ≤ n ≤ 1030.

e) –5 ≤ n ≤ 1. f) –2 ≤ n ≤ 9.

Answer:

```
import java.security.SecureRandom;


public class RandomIntegerAssignment {

    public static void main(String[] args) {

        SecureRandom random = new SecureRandom();


        // a) 2 ≤ n ≤ 6

        int randomA = random.nextInt(5) + 2;


        // b) 4 ≤ n ≤ 50

        int randomB = random.nextInt(47) + 4;


        // c) 0 ≤ n ≤ 7

        int randomC = random.nextInt(8);
        // d) 1000 ≤ n ≤ 1030
        int randomD = random.nextInt(31) + 1000;

        // e) –5 ≤ n ≤ 1
        int randomE = random.nextInt(7) - 5;

        // f) –2 ≤ n ≤ 9
        int randomF = random.nextInt(12) - 2;

        // Printing the results
        System.out.println("a) " + randomA);
        System.out.println("b) " + randomB);
        System.out.println("c) " + randomC);
        System.out.println("d) " + randomD);
        System.out.println("e) " + randomE);
        System.out.println("f) " + randomF);
    }
}
```

6.13 **Write statements that will display a random number from each of the following sets:**

a) 0, 3, 6, 9, 12.

b) 1, 2, 4, 8, 16, 32.

c) 10, 20, 30, 40.

Answer:

```
import java.security.SecureRandom;

public class RandomNumberDisplay {
    public static void main(String[] args) {
        SecureRandom random = new SecureRandom();

        // a) 0, 3, 6, 9, 12
        int[] setA = {0, 3, 6, 9, 12};
        int randomAIndex = random.nextInt(setA.length);
        int randomA = setA[randomAIndex];
        System.out.println("a) " + randomA);

        // b) 1, 2, 4, 8, 16, 32
        int[] setB = {1, 2, 4, 8, 16, 32};
        int randomBIndex = random.nextInt(setB.length);
        int randomB = setB[randomBIndex];
        System.out.println("b) " + randomB);

        // c) 10, 20, 30, 40
        int[] setC = {10, 20, 30, 40};
        int randomCIndex = random.nextInt(setC.length);
        int randomC = setC[randomCIndex];
        System.out.println("c) " + randomC);
    }
}
```

6.14 (Floor and Ceil) Write two methods myFloor and myCeil that take a positive double num variable int myFloor(double num) and int myCeil(double num). The myFloor method takes num and returns the largest integer number that is less than or equal to x. The myCeil function takes num and finds the smallest number that is greater than or equal to x. Do not use any Math class methods. Incorporate this method into an application that sends a double value to the functions and tests their ability to calculate the required output.

Answer:

```
public class FloorCeilExample {
    public static void main(String[] args) {
        double testValue = 7.85;

        int floorResult = myFloor(testValue);
        int ceilResult = myCeil(testValue);

        System.out.printf("Original value: %.2f%n", testValue);
        System.out.printf("Floor result: %d%n", floorResult);
        System.out.printf("Ceil result: %d%n", ceilResult);
    }

    // Method to calculate floor value
    private static int myFloor(double num) {
        int floor = (int) num; // Casting to int truncates the decimal part
        return floor;
    }

    // Method to calculate ceil value
    private static int myCeil(double num) {
        int ceil = (int) num; // Initialize with the truncated value
        if (num > ceil) {
            ceil++; // If there's a decimal part, increment the result
        }
        return ceil;
    }
}
```

6.15 **(Hypotenuse Calculations) Define a method hypotenuse that calculates the hypotenuse of a right triangle when the lengths of the other two sides are given. The method should take two arguments of type double and return the hypotenuse as a double. Incorporate this method into an application that reads values for side1 and side2 and performs the calculation with the hypotenuse method. Use Math methods pow and sqrt to determine the length of the hypotenuse for each of the triangles in Fig. 6.15. [Note: Class Math also provides method hypot to perform this calculation.]**

Answer:

```
import java.util.Scanner;

public class HypotenuseCalculator {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt for side1 and side2
        System.out.print("Enter the length of side1: ");
        double side1 = scanner.nextDouble();

        System.out.print("Enter the length of side2: ");
        double side2 = scanner.nextDouble();

        // Calculate and display the hypotenuse
        double result = hypotenuse(side1, side2);
        System.out.printf("The hypotenuse is: %.2f%n", result);

        scanner.close();
    }

    // Method to calculate the hypotenuse of a right triangle
    private static double hypotenuse(double side1, double side2) {
        // Using Math.pow and Math.sqrt to calculate hypotenuse
        double hypotenuse = Math.sqrt(Math.pow(side1, 2) + Math.pow(side2, 2));
        return hypotenuse;
    }
}
```

6.16 **6.16 (Multiples) Write a method isMultiple that determines, for a pair of integers, whether the second integer is a multiple of the first. The method should take two integer arguments and return true if the second is a multiple of the first and false otherwise. [Hint: Use the remainder operator.] Incorporate this method into an application that inputs a series of pairs of integers (one pair at a time) and determines whether the second value in each pair is a multiple of the first.**

Answer:

```
import java.util.Scanner;


public class MultiplesChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        // Prompt for pairs of integers until the user decides to exit
        while (true) {
            System.out.print("Enter the first integer (or enter -1 to exit): ");
            int firstNumber = scanner.nextInt();


            // Exit the loop if the user enters -1
            if (firstNumber == -1) {
                System.out.println("Exiting the program. Goodbye!");
                break;
            }


            System.out.print("Enter the second integer: ");
            int secondNumber = scanner.nextInt();


            // Check if the second integer is a multiple of the first and display the result
            boolean isMultiple = isMultiple(firstNumber, secondNumber);
            System.out.printf("%d is %s a multiple of %d%n",
                    secondNumber, isMultiple ? "" : "not", firstNumber);
        }
        scanner.close();
    }

    // Method to check if the second integer is a multiple of the first
    private static boolean isMultiple(int first, int second) {
        return second % first == 0;
    }
}
```

6.17 (Divisible by 5) Write a method isDivisible that uses the remainder operator (%) to determine whether ten input integers are divisible by 5 or not. The method should take an integer argument and return true if the integer is divisible by 5 and false otherwise. Incorporate this method into an application that inputs a sequence of integers (one at a time) and determines the result.

Answer:

```
import java.util.Scanner;

public class DivisibleByFiveChecker {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt for integers until the user decides to exit
        while (true) {
            System.out.print("Enter an integer (or enter 0 to exit): ");
            int inputNumber = scanner.nextInt();
            // Exit the loop if the user enters 0
            if (inputNumber == 0) {
                System.out.println("Exiting the program. Goodbye!");
                break;
            }

            // Check if the integer is divisible by 5 and display the result
            boolean divisibleByFive = isDivisible(inputNumber);
            System.out.printf("%d is %sdivisible by 5%n",
                    inputNumber, divisibleByFive ? "" : "not ");
        }

        scanner.close();
    }

    // Method to check if an integer is divisible by 5
    private static boolean isDivisible(int number) {
        return number % 5 == 0;
    }
}
```

6.18 (Displaying a Square of Asterisks) Write a method squareOfAsterisks that displays a solid square (the same number of rows and columns) of asterisks whose side is specified in integer parameter side. For example, if side is 4, the method should display

```
****
****
```

```
****

****
```

Incorporate this method into an application that reads an integer value for side from the user and outputs the asterisks with the squareOfAsterisks method.

Answer:

```
import java.util.Scanner;

public class SquareOfAsterisks {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Prompt the user for the side length of the square
    System.out.print("Enter the side length of the square: ");
    int side = scanner.nextInt();

    // Display the square of asterisks
    squareOfAsterisks(side);

    scanner.close();
  }

  // Method to display a solid square of asterisks
  private static void squareOfAsterisks(int side) {
    for (int row = 1; row <= side; row++) {
            for (int col = 1; col <= side; col++) {
                System.out.print("*");
            }
            System.out.println(); // Move to the next line after each row
        }
    }
}
```

6.19

(Displaying a Square of Any Character) Modify the method created in Exercise 6.18 to receive a second parameter of type char called fillCharacter. Form the square using the char provided as an argument. Thus, if side is 5 and fillCharacter is #, the method should display

```
#####
#####
#####
#####
#####
```

Use the following statement (in which input is a Scanner object) to read a character from the user at the keyboard: char fill = input.next().charAt(0);

Answer:

```
import java.util.Scanner;

public class SquareOfCharacters {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        }

        // Prompt the user for the side length of the square
        System.out.print("Enter the side length of the square: ");
        int side = scanner.nextInt();

        // Prompt the user for the character to fill the square
        System.out.print("Enter the character to fill the square: ");
        char fillCharacter = scanner.next().charAt(0);

        // Display the square of characters
        squareOfCharacters(side, fillCharacter);

        scanner.close();
    }

    // Method to display a solid square of characters
    private static void squareOfCharacters(int side, char fillCharacter) {
        for (int row = 1; row <= side; row++) {
            for (int col = 1; col <= side; col++) {
                System.out.print(fillCharacter);
            }
            System.out.println(); // Move to the next line after each row
        }
    }
}
```

6.20 (Circle Area) Write an application that prompts the user for the radius of a circle and uses a method called circleArea to calculate the area of the circle.

Answer:

```
import java.util.Scanner;

public class CircleAreaCalculator {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Prompt the user for the radius of the circle
    System.out.print("Enter the radius of the circle: ");
    double radius = scanner.nextDouble();

    // Calculate and display the area of the circle
    double area = circleArea(radius);
    System.out.printf("The area of the circle with radius %.2f is: %.2f%n", radius, area);

    scanner.close();
  }

  // Method to calculate the area of a circle
  private static double circleArea(double radius) {
    // Area = π * r^2
```

```
        return Math.PI * Math.pow(radius, 2);

    }

}
```

6.21 **(Beautifying Strings) Write methods that accomplish each of the following tasks:**

a) Check whether the string is terminated by a full stop, and if not, add a full stop.

b) Check whether the string starts with a capital letter, and if not, capitalize the first letter.

c) Use the methods developed in parts (a) and (b) and to write a method beautifyString that receives a string from the user, and then calls the methods in (a) and (b) to make sure that the string is properly formatted, in other words, the string has a full stop at the end, and a capitalized first letter. Make sure you output the string after it has been beautified

Answer:

```
import java.util.Scanner;


public class BeautifyStrings {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);


        // Prompt the user for a string

        System.out.print("Enter a string: ");

        String inputString = scanner.nextLine();


        // Beautify the string and display the result

        String beautifiedString = beautifyString(inputString);

        System.out.println("Beautified String: " + beautifiedString);
        scanner.close();

    }


    // Task (a): Check and add a full stop at the end if not present

    private static String addFullStop(String input) {

        if (!input.endsWith(".")) {

            input += ".";

        }

        return input;

    }


    // Task (b): Check and capitalize the first letter if not already capitalized

    private static String capitalizeFirstLetter(String input) {

        if (input.length() > 0 && !Character.isUpperCase(input.charAt(0))) {

            input = Character.toUpperCase(input.charAt(0)) + input.substring(1);

        }

        return input;

    }


    // Task (c): Beautify the string using methods from (a) and (b)

    private static String beautifyString(String input) {

        // Beautify the string by adding a full stop and capitalizing the first letter

        input = addFullStop(input);

        input = capitalizeFirstLetter(input);
```

```
        return input;
    }
}
```

6.22 (Temperature Conversions) Implement the following integer methods:

a) Method Kelvin returns the Kelvin equivalent of a Celsius temperature, using the calculation Kelvin = Celsius + 273.15;

b) Method Celsius returns the Celsius equivalent of a Kelvin temperature, using the calculation Celsius = Kelvin - 273.15;

c) Use the methods from parts (a) and (b) to write an application that enables the user to enter a Kelvin temperature and display the Celsius equivalent, or, to enter a Celsius temperature and display the Kelvin equivalent.

Answer:

```
import java.util.Scanner;


public class TemperatureConverter {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.println("Choose an option:");
        System.out.println("1. Convert Kelvin to Celsius");
        System.out.println("2. Convert Celsius to Kelvin");


        int option = scanner.nextInt();
    if (option == 1) {
        // Convert Kelvin to Celsius
        System.out.print("Enter the Kelvin temperature: ");
        double kelvinTemperature = scanner.nextDouble();
        double celsiusEquivalent = kelvinToCelsius(kelvinTemperature);
        System.out.printf("%.2f Kelvin is %.2f Celsius%n", kelvinTemperature, celsiusEquivalent);
    } else if (option == 2) {
        // Convert Celsius to Kelvin
        System.out.print("Enter the Celsius temperature: ");
        double celsiusTemperature = scanner.nextDouble();
        double kelvinEquivalent = celsiusToKelvin(celsiusTemperature);
        System.out.printf("%.2f Celsius is %.2f Kelvin%n", celsiusTemperature, kelvinEquivalent);
    } else {
        System.out.println("Invalid option. Please choose 1 or 2.");
    }

    scanner.close();
}

// Method to convert Kelvin to Celsius
private static double kelvinToCelsius(double kelvin) {
    return kelvin - 273.15;
}
```

```
    // Method to convert Celsius to Kelvin
    private static double celsiusToKelvin(double celsius) {
        return celsius + 273.15;
    }
}
```

6.23 (**Find the Minimum) Write a method minimum3 that returns the smallest of three floating point numbers. Use the Math.min method to implement minimum3. Incorporate the method into an application that reads three values from the user, determines the smallest value and displays the result.**

Answer:

```
import java.util.Scanner;

public class MinimumFinder {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter three floating-point numbers
        System.out.print("Enter the first number: ");
        double num1 = scanner.nextDouble();

        System.out.print("Enter the second number: ");
        double num2 = scanner.nextDouble();
        System.out.print("Enter the third number: ");

        double num3 = scanner.nextDouble();


        // Find and display the smallest of the three numbers

        double smallest = minimum3(num1, num2, num3);

        System.out.printf("The smallest number is: %.2f%n", smallest);


        scanner.close();

    }


    // Method to find the smallest of three floating-point numbers

    private static double minimum3(double num1, double num2, double num3) {

        return Math.min(Math.min(num1, num2), num3);

    }

}
```

6.24 **(Perfect Numbers) An integer number is said to be a perfect number if its factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number, because 6 = 1 + 2 + 3. Write a method isPerfect that determines whether parameter number is a perfect number. Use this method in an application that displays all the perfect numbers between 1 and 1000. Display the factors of each perfect number to confirm that the number is indeed perfect. Challenge the computing power of your computer by testing numbers much larger than 1000. Display the results**

Answer:

```
public class PerfectNumbers {
```

```
public static void main(String[] args) {
    System.out.println("Perfect Numbers between 1 and 1000:");


    for (int i = 1; i <= 1000; i++) {
        if (isPerfect(i)) {
            System.out.printf("%d is a perfect number. Factors: ", i);
            displayFactors(i);
        }
    }
}


// Method to determine whether a number is a perfect number
private static boolean isPerfect(int number) {
    int sum = 1; // Start with 1 as a factor


    for (int i = 2; i <= Math.sqrt(number); i++) {
        if (number % i == 0) {
            sum += i;


            // If the factors are distinct, add the corresponding factor
            if (i != number / i) {
                sum += number / i;
            }
        }
    }
```

```
        return sum == number;
    }

    // Method to display factors of a number
    private static void displayFactors(int number) {
        System.out.print("1 ");
        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                System.out.print(i + " ");
                if (i != number / i) {
                    System.out.print(number / i + " ");
                }
            }
        }
        System.out.println();
    }
}
```

6.25 (Prime Numbers) A positive integer is prime if it's divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime, but 4, 6, 8 and 9 are not. The number 1, by definition, is not prime. a) Write a method that determines whether a number is prime. b) Use this method in an application that determines and displays all the prime numbers less than 10,000. How many numbers up to 10,000 do you have to test to ensure that you've found all the primes? c) Initially, you might think that n/2 is the upper limit for which you must test to see whether a number n is prime, but you need only go as high as the square root of n. Rewrite the program and run in both ways.

Answer:

```
public class PrimeNumbers {
    public static void main(String[] args) {
        // Part (b): Display all prime numbers less than 10,000
        System.out.println("Prime Numbers less than 10,000:");
        for (int i = 2; i < 10000; i++) {
            if (isPrime(i)) {
                System.out.print(i + " ");
            }
        }

        // Part (c): Display prime numbers using the square root optimization
        System.out.println("\n\nPrime Numbers (using square root optimization):");
        for (int i = 2; i < 10000; i++) {
            if (isPrimeWithSqrtOptimization(i)) {
                System.out.print(i + " ");
            }
        }
    }

    // Part (a): Method to determine whether a number is prime
    private static boolean isPrime(int number) {
        if (number < 2) {
            return false;
        }

        for (int i = 2; i < number; i++) {
            if (number % i == 0) {
                return false; // Found a factor other than 1 and itself
            }
        }

        return true; // No factors other than 1 and itself
    }

    // Part (c): Method to determine whether a number is prime using the square root optimization
    private static boolean isPrimeWithSqrtOptimization(int number) {
        if (number < 2) {
            return false;
        }

        for (int i = 2; i <= Math.sqrt(number); i++) {
            if (number % i == 0) {
                return false; // Found a factor other than 1 and itself
            }
        }

        return true; // No factors other than 1 and itself
    }
```

```
}
```

6.26 (Calculating the Sum of Digits) Write a method that takes a four-digit integer value and returns the sum of the digits. For example, given the number 7631, the method should return 17. Incorporate the method into an application that reads a value from the user and displays the result.

Answer:

```
import java.util.Scanner;


public class SumOfDigitsCalculator {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);


    // Prompt the user to enter a four-digit integer
    System.out.print("Enter a four-digit integer: ");
    int number = scanner.nextInt();


    // Validate that the entered number is four digits
    if (isValidFourDigitNumber(number)) {
      // Calculate and display the sum of digits
      int digitSum = sumOfDigits(number);
      System.out.printf("The sum of the digits is: %d%n", digitSum);
    } else {
      System.out.println("Invalid input. Please enter a four-digit integer.");
    }
```

```
        scanner.close();
    }

    // Method to calculate the sum of digits of a four-digit number
    private static int sumOfDigits(int number) {
        int sum = 0;

        // Extract each digit and add to the sum
        while (number > 0) {
            sum += number % 10;
            number /= 10;
        }

        return sum;
    }

    // Method to check if a number is a four-digit integer
    private static boolean isValidFourDigitNumber(int number) {
        return number >= 1000 && number <= 9999;
    }
}
```

6.27 (Greatest Common Divisor) The greatest common divisor (GCD) of two integers is the largest integer that evenly divides each of the two numbers. Write a method gcd that returns the greatest common divisor of two integers. [Hint: You might want to use Euclid's algorithm. You can find information about it at en.wikipedia.org/wiki/Euclidean_algorithm.] Incorporate the method into an application that reads two values from the user and displays the result

Answer:

```
import java.util.Scanner;

public class GreatestCommonDivisorCalculator {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);

    // Prompt the user to enter two integers
    System.out.print("Enter the first integer: ");
    int num1 = scanner.nextInt();

    System.out.print("Enter the second integer: ");
    int num2 = scanner.nextInt();

    // Calculate and display the greatest common divisor
    int gcdResult = gcd(num1, num2);
    System.out.printf("The greatest common divisor of %d and %d is: %d%n", num1, num2, gcdResult);

    scanner.close();
  }

  // Method to calculate the greatest common divisor using Euclid's algorithm
  private static int gcd(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return Math.abs(a); // Return the absolute value as GCD is always non-negative
  }
}
```

6.28 **Write a method sportsRecommender that inputs a Celsius temperature and returns “It’s lovely weather for sports today!” if it’s 20–30 °C, “It’s reasonable weather for sports today.” if it’s 10–40 °C, and “Please exercise with care today, watch out for the weather!” otherwise. Create an application to test the method**

Answer:

```
import java.util.Scanner;

public class SportsRecommender {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter the Celsius temperature
        System.out.print("Enter the Celsius temperature: ");
        double temperature = scanner.nextDouble();
        // Get and display the sports recommendation

        String recommendation = sportsRecommender(temperature);

        System.out.println(recommendation);


        scanner.close();

    }


    // Method to provide sports recommendation based on Celsius temperature

    private static String sportsRecommender(double temperature) {

        if (temperature >= 20 && temperature <= 30) {

            return "It’s lovely weather for sports today!";

        } else if (temperature >= 10 && temperature <= 40) {

            return "It’s reasonable weather for sports today.";

        } else {

            return "Please exercise with care today, watch out for the weather!";

        }

      }

}
```

6.29 (**Coin Tossing) Write an application that simulates coin tossing. Let the program toss a coin each time the user chooses the “Toss Coin” menu option. Count the number of times each side of the coin appears. Display the results. The program should call a separate method flip that takes no arguments and returns a value from a Coin enum (HEADS and TAILS). [Note: If the program realistically simulates coin tossing, each side of the coin should appear approximately half the time.]**

Answer:

```
import java.util.Scanner;


public class SportsRecommender {

  public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);


    // Prompt the user to enter the Celsius temperature

    System.out.print("Enter the Celsius temperature: ");

    double temperature = scanner.nextDouble();


    // Get and display the sports recommendation

    String recommendation = sportsRecommender(temperature);

    System.out.println(recommendation);


    scanner.close();

  }


  // Method to provide sports recommendation based on Celsius temperature

  private static String sportsRecommender(double temperature) {

    if (temperature >= 20 && temperature <= 30) {

      return "It’s lovely weather for sports today!";

    } else if (temperature >= 10 && temperature <= 40) {

      return "It’s reasonable weather for sports today.";

    } else {
            return "Please exercise with care today, watch out for the weather!";
        }
    }
}
```

6.30

(Guess the Number) Write an application that plays "guess the number" as follows: Your program chooses the number to be guessed by selecting a random integer in the range 1 to 1000. The application displays the prompt Guess a number between 1 and 1000. The player inputs a first guess. If the player's guess is incorrect, your program should display Too high. Try again. or Too low. Try again. to help the player "zero in" on the correct answer. The program should prompt the user for the next guess. When the user enters the correct answer, display Congratulations. You guessed the number!, and allow the user to choose whether to play again. [Note: The guessing technique employed in this problem is similar to a binary search, which is discussed in Chapter 19, Searching, Sorting and Big O.]

Answer:

```
import java.util.Random;
import java.util.Scanner;

public class GuessTheNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        Random random = new Random();

        do {
            playGuessTheNumber(random);
            // Ask the user if they want to play again
            System.out.print("Do you want to play again? (y/n): ");
        } while (scanner.next().equalsIgnoreCase("y"));

        System.out.println("Thanks for playing! Goodbye!");
        scanner.close();
    }

    // Method to play the "guess the number" game
    private static void playGuessTheNumber(Random random) {
        int numberToGuess = random.nextInt(1000) + 1;
        int guess;
        int attempts = 0;

        System.out.println("Guess a number between 1 and 1000.");

        do {
            // Prompt the user for a guess
            System.out.print("Enter your guess: ");
            guess = getValidGuess();

            // Check if the guess is correct
            if (guess < numberToGuess) {
                System.out.println("Too low. Try again.");
            } else if (guess > numberToGuess) {

                System.out.println("Too high. Try again.");

            } else {

                System.out.printf("Congratulations! You guessed the number in %d attempts!%n", attempts);

            }


            attempts++;

        } while (guess != numberToGuess);

    }


    // Method to get a valid guess from the user

    private static int getValidGuess() {

        Scanner scanner = new Scanner(System.in);

        while (!scanner.hasNextInt()) {

            System.out.print("Invalid input. Enter a valid number: ");

            scanner.next(); // Consume the invalid input

        }

        return scanner.nextInt();

    }

}
```

6.31 (Guess the Number Modification) Modify the program of Exercise 6.30 to count the number of guesses the player makes. If the number is 10 or fewer, display Either you know the secret or you got lucky! If the player guesses the number in 10 tries, display Aha! You know the secret! If the player makes more than 10 guesses, display You should be able to do better! Why should it take no more than 10 guesses? Well, with each "good guess," the player should be able to eliminate half of the numbers, then half of the remaining numbers, and so on.

Answer:

```
import java.util.Random;

import java.util.Scanner;


public class GuessTheNumberModified {

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Random random = new Random();


        do {

            playGuessTheNumber(random);


            // Ask the user if they want to play again

            System.out.print("Do you want to play again? (y/n): ");

        } while (scanner.next().equalsIgnoreCase("y"));


        System.out.println("Thanks for playing! Goodbye!");

        scanner.close();

    }


    // Method to play the "guess the number" game

    private static void playGuessTheNumber(Random random) {

        int numberToGuess = random.nextInt(1000) + 1;
        int guess;

        int attempts = 0;


        System.out.println("Guess a number between 1 and 1000.");


        do {

            // Prompt the user for a guess

            System.out.print("Enter your guess: ");

            guess = getValidGuess();

            attempts++;


            // Check if the guess is correct

            if (guess < numberToGuess) {

               System.out.println("Too low. Try again.");

            } else if (guess > numberToGuess) {

               System.out.println("Too high. Try again.");

            } else {

               System.out.printf("Congratulations! You guessed the number in %d attempts!%n", attempts);


               // Provide additional feedback based on the number of attempts

               if (attempts <= 10) {

                  System.out.println("Either you know the secret or you got lucky!");

               } else if (attempts == 10) {

                  System.out.println("Aha! You know the secret!");

               } else {
```

```
                    System.out.println("You should be able to do better! Why should it take no more than 10 
guesses?");

                }

            }


        } while (guess != numberToGuess);

    }


    // Method to get a valid guess from the user

    private static int getValidGuess() {

        Scanner scanner = new Scanner(System.in);

        while (!scanner.hasNextInt()) {

            System.out.print("Invalid input. Enter a valid number: ");

            scanner.next(); // Consume the invalid input

        }

        return scanner.nextInt();

    }

}
```

6.32 **(Distance Between Points) Write method distance to calculate the distance between two points (x1, y1) and (x2, y2). All numbers and return values should be of type double. Incorporate this method into an application that enables the user to enter the coordinates of the points.**

Answer:

```
import java.util.Scanner;
```

```
public class DistanceBetweenPoints {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Prompt the user to enter the coordinates of the first point
        System.out.print("Enter the x-coordinate of the first point: ");
        double x1 = scanner.nextDouble();
        System.out.print("Enter the y-coordinate of the first point: ");
        double y1 = scanner.nextDouble();

        // Prompt the user to enter the coordinates of the second point
        System.out.print("Enter the x-coordinate of the second point: ");
        double x2 = scanner.nextDouble();
        System.out.print("Enter the y-coordinate of the second point: ");
        double y2 = scanner.nextDouble();

        // Calculate and display the distance between the points
        double distanceResult = distance(x1, y1, x2, y2);
        System.out.printf("The distance between the points is: %.2f%n", distanceResult);

        scanner.close();
    }

    // Method to calculate the distance between two points
    private static double distance(double x1, double y1, double x2, double y2) {
        return Math.sqrt(Math.pow(x2 - x1, 2) + Math.pow(y2 - y1, 2));
    }
}
```

6.34 **(Table of Binary, Octal and Hexadecimal Numbers) Write an application that displays a table of the binary, octal and hexadecimal equivalents of the decimal numbers in the range 1 through 256. If you aren't familiar with these number systems, read online Appendix J first.**

Answer:

```
public class NumberConversionTable {
    public static void main(String[] args) {
        // Display header
        System.out.printf("%-10s%-12s%-12s%n", "Decimal", "Binary", "Hexadecimal");

        // Display number conversions for the range 1 through 256
        for (int decimal = 1; decimal <= 256; decimal++) {
            String binary = Integer.toBinaryString(decimal);
            String hexadecimal = Integer.toHexString(decimal);

            // Display the formatted row
            System.out.printf("%-10d%-12s%-12s%n", decimal, binary, hexadecimal);
        }
    }
}
```