Coding Exercises

1. Hello Coding
2. Hello World
3. Time Converter

In Java, you have been given a partially implemented TimeConverter class that aims to convert time from hours or days to minutes. The TimeConverter class has two static methods that need to be completed:

* convertHoursToMinutes(int hours): This method should take an integer value representing hours and return the equivalent value in minutes.
* convertDaysToMinutes(int days): This method should take an integer value representing days and return the equivalent value in minutes.
* **Instructions**
* **Task 1**: Complete the convertHoursToMinutes method in the TimeConverter class so that it accurately converts hours into minutes. The method should return an integer value representing the total number of minutes.
* **Task 2**: Complete the convertDaysToMinutes method in the TimeConverter class so that it accurately converts days into minutes. The method should return an integer value representing the total number of minutes.
* **NOTE:**Return**-1** for invalid cases (days and hours less than 0).
* In Java, you have been given a partially implemented TimeConverter class that aims to convert time from hours or days to minutes. The TimeConverter class has two static methods that need to be completed:
* convertHoursToMinutes(int hours): This method should take an integer value representing hours and return the equivalent value in minutes.
* convertDaysToMinutes(int days): This method should take an integer value representing days and return the equivalent value in minutes.
* **Instructions**
* **Task 1**: Complete the convertHoursToMinutes method in the TimeConverter class so that it accurately converts hours into minutes. The method should return an integer value representing the total number of minutes.
* **Task 2**: Complete the convertDaysToMinutes method in the TimeConverter class so that it accurately converts days into minutes. The method should return an integer value representing the total number of minutes.
* **NOTE:**Return**-1** for invalid cases (days and hours less than 0).

package com.jb.T00CodingExercises;

public class C03TimeConverter

{

        public static int convertHoursToMinutes(int hours)

        {

            return hours\*60;

        }

        public static int convertDaysToMinutes(int days)

        {

            if(days<0)

            {

                return -1;

            }

            return days\*24\*60;

        }

        @SuppressWarnings("unused")

        public static void main(String[] args)

        {

            C03TimeConverter hoursCon = new C03TimeConverter();

            C03TimeConverter dayCon = new C03TimeConverter();

            System.out.println(C03TimeConverter.convertHoursToMinutes(2));

            System.out.println(C03TimeConverter.convertDaysToMinutes(2));

    }

}

// 120

// 2880

1. Exam Result Checker

You have been provided with an incomplete Java class named ExamResult. This class has a method isPass(int marks) which needs to be completed.

The isPass(int marks) method should take an integer value representing the marks obtained by a student in an exam and determine whether the student has passed or failed the exam.

In this case, a student is considered to have **passed** the exam if they have **scored more than 50 marks**. If the student's marks are 50 or less, they are considered to have failed.

**Instructions**

1. Complete the isPass(int marks) method such that it returns a boolean value indicating whether the student has passed or not. This method should return true if the marks are greater than 50 (indicating the student has passed), and false otherwise (indicating the student has failed).
2. Be sure to consider edge cases such as a student scoring exactly 50 marks.

Remember, the goal is to check if a student has passed or failed based on their marks. Good luck!

package com.jb.T00CodingExercises;

public class C04ExamResult {

    public boolean isPass(int marks)

    {

        if (marks<=50)

        {

            return false;

        }

        else return true;

    }

    public static void main(String[] args)

    {

        C04ExamResult eC04ExamResult=new C04ExamResult();

        System.out.println(eC04ExamResult.isPass(50));

        System.out.println(eC04ExamResult.isPass(28));

        System.out.println(eC04ExamResult.isPass(98));

    }

}

// false

// false

// true

1. Sum of Square of First N Numbers

You are given an integer n. Your task is to implement a method calculateSumOfSquares in the class SumOfSquares that calculates and returns the sum of squares of all positive integers up to n (inclusive).

For example, if n = 3, the method should return the sum 1^2 + 2^2 + 3^2 = 14.

If n is less than 0, the method should return -1.

**Instructions**

The incomplete code provided contains a class named SumOfSquares with a method signature public long calculateSumOfSquares(int n).

Complete this method to calculate the sum of squares of all positive integers up to n. If n is less than 0, return -1.

Do not print anything in your code. The method should return the sum or -1.

public class SumOfSquares {

public long calculateSumOfSquares(int n) {

// write your code here

}

}

Good luck!

package com.jb.T00CodingExercises;

public class C05SumOfSquares {

    public long calculateSumOfSquares(int n)

    {

        long sum =0;

        if(n>=0)

        {

            for(int i=0;i<=n;i++)

            {

                long square=i\*i;

                sum=sum+square;

            }

            return sum;

        }

        else return -1;

    }

    public static void main(String[] args)

    {

        C05SumOfSquares sumOfSquares = new C05SumOfSquares();

        System.out.println(sumOfSquares.calculateSumOfSquares(3)); //14

    }

}

1. Dimensions – Feet Inches ; 1 feet = 12 inches

In this problem, you have to complete the implementation of a Dimension class in Java that represents measurements in feet and inches.

Here is the skeleton of the Dimension class:

public class Dimension {

private int feet;

private int inches;

public Dimension(int inches) {

// TODO: Convert the total inches into feet and inches. One foot is 12 inches.

// Store the feet and inches in their respective instance variables.

}

public int getFeet() {

// TODO: Return the value of feet.

return -1;

}

public int getInches() {

// TODO: Return the value of inches.

return -1;

}

}

**Task**

Your task is to implement the Dimension(int inches) constructor and getFeet() & getInches() methods in the Dimension class.

Dimension(int inches) - This constructor will take one parameter, inches (an integer), and it should convert the given inches into feet and inches. If the input inches are less than 0, set feet and inches to -1 to indicate invalid input. One foot is equal to 12 inches.

getFeet() - This method should return the calculated feet from the constructor.

getInches() - This method should return the remaining inches (less than 12) from the constructor.

**Examples**

Here are some examples of how your Dimension class should behave once implemented:

Dimension dim = new Dimension(25);

System.out.println(dim.getFeet()); // Output: 2

System.out.println(dim.getInches()); // Output: 1

In the above example, 25 inches is equal to 2 feet and 1 inch.

Dimension dim = new Dimension(-1);

System.out.println(dim.getFeet()); // Output: -1

System.out.println(dim.getInches()); // Output: -1

In the above example, -1 inch is an invalid input so both getFeet() and getInches() return -1.

Good luck!

package com.jb.T00CodingExercises;

public class C06DimensionFeetInches {

    private int inches;

    private int feet=0;;

        public C06DimensionFeetInches(int inches)

        {

            this.inches=inches;

        }

    public int getFeet()

    {

       if(inches>=0)

       {

        feet=inches/12;

        return feet;

       }

       else return -1;

    }

    public int getInches()

    {

        if(inches>=0)

        {

            int remInchOFFoot=0;

            remInchOFFoot=inches%12;

            return remInchOFFoot;

        }

        else return -1;

    }

    public static void main(String[] args) {

        C06DimensionFeetInches dimensionFeetInches=new C06DimensionFeetInches(25);

        System.out.println(dimensionFeetInches.getFeet()); //2

        System.out.println(dimensionFeetInches.getInches()); //1

    }

}

1. Area of Square and Perimeter

In this exercise, you're tasked to implement a class called Square in Java. This class is responsible for modeling a geometric shape - a square.

A Square object has one private instance variable, side (of type int), which denotes the length of a side of the square.

The Square class has a constructor that takes an integer argument. This argument should be used to initialize the side attribute.

This class also includes two methods:

calculateArea(): This method calculates and returns the area of the square. The area of a square is given by the formula: side \* side. However, in the real world, a square cannot have a side length of zero or less. So, if side is less than or equal to zero, this method should return -1 to indicate an invalid value.

calculatePerimeter(): This method calculates and returns the perimeter of the square. The perimeter of a square is given by the formula: 4 \* side. Similar to the above, if side is less than or equal to zero, this method should return -1 to indicate an invalid value.

Your task is to complete the implementation of the constructor and these two methods.

**Instructions**

Implement the Square constructor. This constructor should take an integer argument and initialize the side attribute with the value of this argument.

Implement the calculateArea() method. If the side attribute is less than or equal to zero, return -1. Otherwise, calculate and return the area of the square using the formula: side \* side.

Implement the calculatePerimeter() method. If the side attribute is less than or equal to zero, return -1. Otherwise, calculate and return the perimeter of the square using the formula: 4 \* side.

**Example**

Let's consider the following examples:

In the first example, we create a Square object with side equal to 5. Therefore, the calculateArea() method returns 25 (5 \* 5) and the calculatePerimeter() method returns 20 (4 \* 5).

Square square = new Square(5);

System.out.println(square.calculateArea()); // prints: 25

System.out.println(square.calculatePerimeter()); // prints: 20

In the second example, we create a Square object with side equal to 0. Therefore, both the calculateArea() and calculatePerimeter() methods return -1 as they represent invalid values. This is because a square cannot have a side length of zero.

Square squareWithZeroSide = new Square(0);

System.out.println(squareWithZeroSide.calculateArea()); // prints: -1

System.out.println(squareWithZeroSide.calculatePerimeter()); // prints: -1

In the third example, we create a Square object with a non-positive side (-5). Therefore, both the calculateArea() and calculatePerimeter() methods return -1 as they represent invalid values. This is because a square cannot have a side length less than zero.

Square squareWithNonPositiveSide = new Square(-5);

System.out.println(squareWithNonPositiveSide.calculateArea()); // prints: -1

System.out.println(squareWithNonPositiveSide.calculatePerimeter()); // prints: -1

package com.jb.T00CodingExercises;

public class C07SquareAreaPerimeter {

    private int side;

    public C07SquareAreaPerimeter(int side) {

        this.side=side;

    }

    public int calculateArea() {

        if(side>0)

        {

            return side\*side;

        }

        else return -1;

    }

    public int calculatePerimeter() {

        if(side>0)

        {

            return 4\*side;

        }

        else return -1;

    }

    public static void main(String[] args) {

        C07SquareAreaPerimeter squareAreaPerimeter=new C07SquareAreaPerimeter(5);

        System.out.println(squareAreaPerimeter.calculateArea()); //25

        System.out.println(squareAreaPerimeter.calculatePerimeter()); //20

    }

}

1. Distance between 2 points

In this exercise, you have to complete the implementation of the Point class in Java. A point is an entity that has an x-coordinate and a y-coordinate in a two-dimensional space.

Here is the partially completed Point class that you will work with:

// Defining a Point class to represent a point in 2-dimensional space

public class Point {

// x-coordinate of the point

private int x;

// y-coordinate of the point

private int y;

// Constructor for the Point class

public Point(int x, int y) {

this.x = x; // Assigning x-coordinate of the point

this.y = y; // Assigning y-coordinate of the point

}

// Method to get the x-coordinate of the point

public int getX() {

return x;

}

// Method to get the y-coordinate of the point

public int getY() {

return y;

}

// TODO: Implement the method to move the point by dx and dy in x and y direction respectively

public void move(int dx, int dy) {

// Your code here

}

// TODO: Implement the method to calculate the distance from this point to another point

public double distanceTo(Point other) {

// Your code here

}

}

**Task**

Your task is to complete the implementation of the following methods in the Point class:

1. **Method: move(int dx, int dy)**

This method adjusts the current position of the point in the 2-dimensional space. The parameters dx and dy represent the changes in x-coordinate and y-coordinate respectively.

After calling this method, the new coordinates of the point should be (current x + dx, current y + dy).

Example: If we have a point at (3, 4) and call the move method with arguments 1 and 2, the new coordinates of the point should be (3+1, 4+2) i.e., (4, 6).

1. **Method: distanceTo(Point other)**

This method calculates and returns the Euclidean distance between the current point and another point other.

The Euclidean distance d between two points (x1, y1) and (x2, y2) is calculated as: d = sqrt((x2-x1)^2 + (y2-y1)^2).

Example: If we have a point at (3, 4) and want to calculate the distance to another point at (6, 8), the distance should be sqrt((6-3)^2 + (8-4)^2) i.e., sqrt(9+16) i.e., sqrt(25) i.e., 5.0.

TIP: Use the Math.sqrt function to compute the square root of a number.

**Example**

Once your code is implemented correctly, the following example should work:

Point p1 = new Point(3, 4);

Point p2 = new Point(6, 8);

p1.move(1, 1); // After this move, the point p1 should be at (4, 5)

double distance = p1.distanceTo(p2); // The distance from p1 to p2 should be sqrt((6-4)^2 + (8-5)^2) = sqrt(4+9) = sqrt(13) ≈ 3.60555

package com.jb.T00CodingExercises;

public class C08DistanceBw2Points {

    private int x;

    private int y;

     int y2;

    public C08DistanceBw2Points(int x, int y) {

        this.x = x;  // Assigning x-coordinate of the point

        this.y = y;  // Assigning y-coordinate of the point

    }

    public int getX() {

        return x;

    }

    public int getY() {

        return y;

    }

    public void move(int dx, int dy)

    {

        this.x=this.x+dx;

        this.y=this.y+dy;

        System.out.println(x+","+y);

    }

    public double distanceTo(C08DistanceBw2Points other)

    {

        int diffx=this.x-other.x;

        int diffy=this.y-other.y;

        return Math.sqrt(diffx\*diffx+diffy\*diffy);

    }

    public static void main(String[] args) {

        C08DistanceBw2Points distanceBw2Points1=new C08DistanceBw2Points(3, 4);

        C08DistanceBw2Points distanceBw2Points2=new C08DistanceBw2Points(6, 8);

        System.out.println(distanceBw2Points1.distanceTo(distanceBw2Points2)); //5.0

        distanceBw2Points1.move(1, 1); //4,5

        System.out.println(distanceBw2Points1.distanceTo(distanceBw2Points2)); //3.605551275463989

    }

}

1. RGB Color Invert

The **Red-Green-Blue (RGB) model** is a popular way to represent colors in computer systems. In this model, each color is represented as a combination of the primary colors red, green, and blue. Each of these primary colors can have intensity values ranging from 0 to 255.

In this exercise, you are tasked with completing the RGBColor class that models a color in the RGB model. The class has fields for the red, green, and blue intensities, and methods to get these values.

Additionally, there's an invert method that changes the color to its complementary color (the inverse color on a color wheel).

You need to implement the constructor, the getter methods for the red, green, and blue values, and the invert method.

The invert method should subtract the current intensity value of each primary color from 255 to get the inverted color.

Please follow the provided skeleton code and fill in the missing parts.

**Instructions**

Start by implementing the constructor of the RGBColor class. The constructor should take three arguments: red, green, and blue. These should be used to initialize the red, green, and blue fields of the class.

Implement the getRed, getGreen, and getBlue methods. These methods should return the current values of the red, green, and blue fields respectively.

Implement the invert method. This method should subtract the current value of each of the red, green, and blue fields from 255, and assign the result back to the respective field.

**Examples**

Below are three examples of creating an RGBColor object, getting the red, green, and blue values, and inverting the color.

**Example 1**

RGBColor color = new RGBColor(255, 0, 0);

System.out.println(color.getRed()); // Prints: 255

System.out.println(color.getGreen()); // Prints: 0

System.out.println(color.getBlue()); // Prints: 0

color.invert();

System.out.println(color.getRed()); // Prints: 0

System.out.println(color.getGreen()); // Prints: 255

System.out.println(color.getBlue()); // Prints: 255

**Example 2**

RGBColor color = new RGBColor(0, 255, 0);

color.invert();

System.out.println(color.getGreen()); // Prints: 0

**Example 3**

RGBColor color = new RGBColor(128, 128, 128);

color.invert();

System.out.println(color.getRed()); // Prints: 127

System.out.println(color.getGreen()); // Prints: 127

System.out.println(color.getBlue()); // Prints: 127

package com.jb.T00CodingExercises;

// RGBColor class representing a color using Red Green Blue (RGB) model.

public class C09RGBColorInvert {

    // The Red, Green, Blue color values range from 0 to 255.

    private int red;

    private int green;

    private int blue;

    // Constructor for RGBColor class which initializes the color with provided red, green and blue values.

    public C09RGBColorInvert(int red, int green, int blue) {

        this.red=red;

        this.green=green;

        this.blue=blue;

    }

    // Getter method to get the red value of the color.

    public int getRed()

    {

        return this.red;

    }

    // Getter method to get the green value of the color.

    public int getGreen() {

        return this.green;

    }

    // Getter method to get the blue value of the color.

    public int getBlue() {

        return this.blue;

    }

    // Method to invert the color. The inversion is done by subtracting each color component from 255.

    public void invert() {

        this.red=255-this.red;

        this.green=255-this.green;

        this.blue=255-this.blue;

        System.out.println(red+","+green+","+blue);

    }

    public static void main(String[] args) {

        C09RGBColorInvert rC09rgbColorInvert=new C09RGBColorInvert(255, 0, 0);

        rC09rgbColorInvert.invert();//255,0,0 -> 0,225,225

    }

}

1. Check Triangle or Not

**Coding Exercise: Is Valid Triangle**

You are given three integer inputs that represent the angles of a triangle. You need to complete a method named isValidTriangle(int angle1, int angle2, int angle3), inside a class named TriangleValidator. This method should return a boolean value – true if the given angles form a valid triangle, and false if they do not.

The conditions for a valid triangle are:

Each of the angles must be a positive integer.

The sum of all the three angles must be exactly 180.

Please complete the isValidTriangle method in the provided TriangleValidator class following the template:

public class TriangleValidator {

public boolean isValidTriangle(int angle1, int angle2, int angle3) {

// write your code here

return false;

}

}

**Details**

Ensure you return false if any of the input angles are non-positive (i.e., less than or equal to zero).

Compute the sum of the angles and check if it equals 180.

Return true if all conditions are met, false otherwise.

Good luck!

package com.jb.T00CodingExercises;

public class C10TriangleValidator

{

    public boolean isValidTriangle(int angle1, int angle2, int angle3)

    {

        int tri=0;

        if(angle1>0 && angle2>0 && angle3>0)

        {

            tri=angle1+angle2+angle3;

            if (tri==180)

            {

                return true;

            }

        }

        return false;

    }

    public static void main(String[] args) {

        C10TriangleValidator triangleValidator= new C10TriangleValidator();

        System.out.println(triangleValidator.isValidTriangle(12,90,90));

        System.out.println(triangleValidator.isValidTriangle(60,60,60));

    }

}

// false

// true

1. Right Angle Triangle Validator

You have been given a class TriangleValidator with an incomplete method isRightAngled(int side1, int side2, int side3). This method should take three integer values that represent the lengths of the sides of a triangle. Your task is to complete the method so it can accurately determine if the triangle with the given sides is a right-angled triangle. The method should return true if the triangle is right-angled and false otherwise.

In a right-angled triangle, the square of the length of the hypotenuse (the side opposite the right angle) is equal to the sum of the squares of the lengths of the other two sides. This is known as the Pythagorean theorem.

If any side has a non-positive length, it's not considered a valid triangle.

**Instructions**

Start with the TriangleValidator class. Inside this class, you'll find the method isRightAngled(int side1, int side2, int side3).

Your first task is to validate the lengths of the sides. If any side length is less than or equal to zero, return false.

Next, apply the Pythagorean theorem to determine if the triangle is right-angled. Remember to check all combinations of sides.

If the theorem holds true for any combination of sides, return true. This indicates that the triangle is right-angled.

If the theorem doesn't hold true for any combination of sides, return false. This indicates that the triangle is not right-angled.

**Good luck!**

package com.jb.T00CodingExercises;

public class C11RightAngleTriangleValidator {

    public boolean isRightAngled(int side1, int side2, int side3)

    {

        if(side1<=0 || side2<=0 || side3<=0)

        {

            return false;

        }

            int lhs1=side1\*side1+side2\*side2;

            int lhs2=side2\*side2+side3\*side3;

            int lhs3=side3\*side3+side1\*side1;

            int rhs1=side3\*side3;

            int rhs2=side1\*side1;

            int rhs3=side2\*side2;

            if(lhs1==rhs1)

            {

                return true;

            }

            if(lhs2==rhs2)

            {

                return true;

            }

            if(lhs3==rhs3)

            {

                return true;

            }

            return false;

    }

    public static void main(String[] args) {

        C11RightAngleTriangleValidator c11RightAngleTriangleValidator=new C11RightAngleTriangleValidator();

        System.out.println(c11RightAngleTriangleValidator.isRightAngled(3, 4, 6));

        System.out.println(c11RightAngleTriangleValidator.isRightAngled(3, 4, 5));

    }

}

// false

// true

1. Leap Year

You are given a class LeapYearChecker with a method isLeapYear(int year). Your task is to complete the implementation of the method to determine if a given year is a leap year.

The rules are a little tricky. Read them carefully.

A year is a leap year in the Gregorian calendar if:

* It is divisible by 4 (AND)
* It is NOT divisible by 100 (except when it is divisible by 400)

Not Divisible by 4 - NOT Leap Year (2041)

Divisible by 4 and NOT divisible by 100 - Leap Year (2048)

Divisible by 4 and divisible by 100 - Additional check needed

Divisible by 4, divisible by 100, divisible by 400 - Leap Year (2000, 2400)

Divisible by 4, divisible by 100, NOT divisible by 400 - NOT Leap Year (2100, 2200, 2300)

* + **Instructions**
* Implement the isLeapYear method to return true if the year is a leap year and false otherwise.
* You need to account for the edge case where the year is less than 1. For such a case, the method should return false.
* public class LeapYearChecker {
* public boolean isLeapYear(int year) {
* // Implement your code here
* }
* }
  + Please ensure your code passes all the test cases, including edge cases, and adheres to the above guidelines.

package com.jb.T00CodingExercises;

public class C12LeapYearChecker {

    public boolean isLeapYear(int year) {

        if(year<=0)

        {

            return false;

        }

        if(year % 4 != 0) // not divisible by 4 is not leap year

        {

            return false;

        }

        if(year % 4 == 0 && year %100 != 0) // divisible by 4 and not divisible by 100 is leap year

        {

            return true;

        }

        if(year % 4 == 0 && year %100 == 0) // divisible by 4 and divisible by 100 then divisible by 400 is leap year

        {

            if (year%400==0)

            {

                return true;

            }

            else return false;

        }

       return false;

    }

    public static void main(String[] args) {

        C12LeapYearChecker c12LeapYearChecker=new C12LeapYearChecker();

        System.out.println(c12LeapYearChecker.isLeapYear(2048));

    }

}

//true

1. Perfect Number Checker

Sum of exceeding factors of that number = that number

6=1,2,3,=6

In this exercise, your task is to implement a method in the PerfectNumberChecker class that checks whether a number is a "perfect number".

A **Perfect Number** is a positive integer that is equal to the **sum of all of its positive divisors**, excluding itself.

For example, the number 6 is a perfect number because its divisors are 1, 2, and 3. The sum of these divisors is equal to 6, so 6 is a perfect number. Similarly, 28 is also a perfect number because its divisors (1, 2, 4, 7, 14) sum up to 28.

1. public class PerfectNumberChecker {
2. public boolean isPerfectNumber(int number) {
3. // Write code here
4. }
5. }

**Instructions**

1. The isPerfectNumber(int number) method should return true if the number is a perfect number, and false otherwise.
2. A perfect number is always a positive integer. If the number is zero or a negative number, return false.
3. You can assume that the input number will always be an integer.

package com.jb.T00CodingExercises;

public class C13PerfectNumberChecker

{

    public boolean isPerfectNumber(int number) {

        if(number<=0)

        {

            return false;

        }

        int perfect=0;

        for(int i=1;i<number;i++)

        {

            if(number%i==0)

            {

                perfect=perfect+i;

            }

            if(perfect==number)

            {

                return true;

            }

        }

        return false;

    }

    public static void main(String[] args) {

        C13PerfectNumberChecker c13PerfectNumberChecker=new C13PerfectNumberChecker();

        System.out.println(c13PerfectNumberChecker.isPerfectNumber(6));

    }

}

//true

1. Student Grade

You are tasked with implementing the functionality of a Student class which calculates and assigns the student's grade based on the provided marks.

A student's grade is determined by the following criteria:

* If the student's marks are less than 0 or greater than 100, the grade should be 'X'.
* If the student's marks are greater than or equal to 90, the grade should be 'A'.
* If the student's marks are greater than or equal to 80 but less than 90, the grade should be 'B'.
* If the student's marks are greater than or  equal to 70 but less than 80, the grade should be 'C'.
* If the student's marks are greater than or equal to 60 but less than 70, the grade should be 'D'.
* If the student's marks are greater than or equal to 50 but less than 60, the grade should be 'E'.
* If the student's marks are less than 50, the grade should be 'F'.

**Class Structure**

The class Student contains the following:

* An instance variable marks of type int.
* A constructor that accepts an int parameter to initialize marks.
* A method assignGrade() that returns a char, representing the student's grade based on the provided marks.

Here's the skeleton of the class you will work with:

1. public class Student {
2. private int marks;
4. public Student(int marks) {
5. // TODO: Assign 'marks' to the instance variable 'this.marks'
6. }
8. public char assignGrade() {
9. // TODO: Implement the method which assigns the grade based on marks.
10. // If marks are less than 0 or greater than 100, return 'X'
11. // If marks are greater than or equal to 90, return 'A'
12. // If marks are greater than or equal to 80, return 'B'
13. // If marks are greater than or equal to 70, return 'C'
14. // If marks are greater than or equal to 60, return 'D'
15. // If marks are greater than or equal to 50, return 'E'
16. // If marks are less than 50, return 'F'
17. }
18. }

**Task**

Your task is to implement the Student(int marks) constructor and assignGrade() method in the Student class.

* Student(int marks) - This constructor will take one parameter, marks (an integer), and it should assign it to the marks instance variable.
* assignGrade() - This method should return a character representing the grade of the student, based on the marks they have received.

If marks are:

* less than 0 or greater than 100, return 'X'
* greater than or equal to 90, return 'A'
* greater than or equal to 80, return 'B'
* greater than or equal to 70, return 'C'
* greater than or equal to 60, return 'D'
* greater than or equal to 50, return 'E'
* less than 50, return 'F'

**Examples**

Here are some examples of how your Student class should behave once implemented:

1. Student student = new Student(85);
2. System.out.println(student.assignGrade()); // Should print 'B'
3. Student student = new Student(101);
4. System.out.println(student.assignGrade()); // Should print 'X'

In the first example, the student's marks are 85, so the grade is 'B'. In the second example, the student's marks exceed the maximum possible mark of 100, so the method returns 'X' to indicate an invalid mark.

Good luck! Your effective implementation can pass all the examples and additional tests.

package com.jb.T00CodingExercises;

public class C14StudentGrade {

    private int marks;

    public C14StudentGrade(int marks) {

        this.marks=marks;

    }

    public char assignGrade()

    {

        if(marks<0 || marks>100)

        {

            return 'X';

        }

        else if(marks>=90)

        {

            return 'A';

        }

        else if(marks>=80 && marks<90)

        {

            return 'B';

        }

        else if(marks>=70 && marks<80)

        {

            return 'C';

        }

        else if(marks>=60 && marks<70)

        {

            return 'D';

        }

        else if(marks>=50 && marks<60)

        {

            return 'E';

        }

        else

        {

            return 'F';

        }

    }

    public static void main(String[] args) {

        C14StudentGrade studentGrade=new C14StudentGrade(99);

        System.out.println(studentGrade.assignGrade());

    }

}

//A

1. WeatherAdviser

In this problem, you have to complete the implementation of a WeatherAdviser class in Java that provides advice on what to wear based on the current temperature.

Here is the skeleton of the WeatherAdviser class:

* public class WeatherAdviser {
* public String provideWeatherAdvisory(int temperature) {
* // TODO: Provide a weather advisory message based on the temperature.
* return "";
* }
* }

**Task**

Your task is to implement the provideWeatherAdvisory(int temperature) method in the WeatherAdviser class.

* provideWeatherAdvisory(int temperature) - This method will take one parameter, temperature (an integer), and it should return a String containing advice on what to wear according to the following guidelines:
  + If the temperature is less than 0, return "It's freezing! Wear a heavy coat."
  + If the temperature is between 0 and 10, inclusive, return "It's cold! Bundle up."
  + If the temperature is between 11 and 20, inclusive, return "It's cool! A light jacket will do."
  + If the temperature is above 20, return "It's warm! Enjoy the day."

**Examples**

Here are some examples of how your WeatherAdviser class should behave once implemented:

WeatherAdviser adviser = new WeatherAdviser();

System.out.println(adviser.provideWeatherAdvisory(-5));

// Output: "It's freezing! Wear a heavy coat."

System.out.println(adviser.provideWeatherAdvisory(5));

// Output: "It's cold! Bundle up."

System.out.println(adviser.provideWeatherAdvisory(15));

// Output: "It's cool! A light jacket will do."

System.out.println(adviser.provideWeatherAdvisory(25));

// Output: "It's warm! Enjoy the day."

Good luck!

package com.jb.T00CodingExercises;

public class C15WeatherAdviser {

    public String provideWeatherAdvisory(int temperature)

    {

        if(temperature<0)

        {

            return "It's freezing! Wear a heavy coat.";

        }

        else if(temperature>=0 && temperature <=10)

        {

            return"It's cold! Bundle up.";

        }

        else if(temperature>=11 && temperature <=20)

        {

            return"It's cool! A light jacket will do.";

        }

        else return "It's warm! Enjoy the day.";

    }

    public static void main(String[] args) {

        C15WeatherAdviser c15WeatherAdviser=new C15WeatherAdviser();

        System.out.println(c15WeatherAdviser.provideWeatherAdvisory(11));

    }

}

//It's cool! A light jacket will do.

1. D
2. dd
3. D
4. D
5. D
6. D
7. d