PROJECT 4

# Part 1

## Forecast.java

|  |
| --- |
| package *com.valencia*;  import *java.io.FileNotFoundException*; import *java.io.FileWriter*; import *java.io.IOException*;  public class *Forecast* {  private final double DEFAULT\_TEMPERATURE = 72;  private final *String* DEFAULT\_SKY\_CONDITION = "clear";  private final int DEFAULT\_CHANCE\_OF\_RAIN = 0;  private final *String* RESULT\_FILE = "forecast\_result.txt";   private double temperature;  private *String* skyCondition;  private int chanceOfRain;   public Forecast(double *temperature*, *String skyCondition*, int *chanceOfRain*) {  setTemperature(*temperature*);  setSkyCondition(*skyCondition*);  setChanceOfRain(*chanceOfRain*);  }   public Forecast() {  setTemperature();  setSkyCondition();  setChanceOfRain();  }   public static double convertFahrenheitToCelsius(double *fahrenheit*) {  return (*fahrenheit* - 32) \* ((double) 5 / 9);  }   public static double convertCelsiusToFahrenheit(double *celcius*) {  return *celcius* \* ((double) 9 / 5) + 32;  }   public static double convertFahrenheitToKelvin(double *fahrenheit*) {  return *convertFahrenheitToCelsius*(*fahrenheit*) + 273;  }   public static double convertKelvinToFahrenheit(double *kelvin*) {  return *convertCelsiusToFahrenheit*(*kelvin* - 273);  }   public double getTemperature() {  return temperature;  }   public void setTemperature(double *temperature*) {  if (*temperature* >= -100 && *temperature* <= 150) {  this.temperature = *temperature*;  } else {  setTemperature();  }  }   public void setTemperature() {  this.temperature = DEFAULT\_TEMPERATURE;  }   public *String* getSkyCondition() {  return skyCondition;  }   public void setSkyCondition(*String skyCondition*) {  this.skyCondition = *skyCondition*;  }   public void setSkyCondition() {  this.skyCondition = DEFAULT\_SKY\_CONDITION;  }   public int getChanceOfRain() {  return chanceOfRain;  }   public void setChanceOfRain(int *chanceOfRain*) {  if (*chanceOfRain* >= 0 && *chanceOfRain* <= 100) {  this.chanceOfRain = *chanceOfRain*;  } else {  setChanceOfRain();  }  }   public void setChanceOfRain() {  this.chanceOfRain = DEFAULT\_CHANCE\_OF\_RAIN;  }   public boolean isRaining() {  return skyCondition.equals("raining");  }   public void print() {  *System*.out.printf("Temperature: %.2f °F\n", temperature);  *System*.out.printf("Sky condition: %s\n", skyCondition);  *System*.out.printf("Chance of rain: %d%%\n", chanceOfRain);  }   public void printToFile() {  try (*FileWriter* writer = new FileWriter(RESULT\_FILE)) {  writer.write(*String*.*format*("Temperature: %.2f °F", temperature));  writer.write(*System*.*getProperty*("line.separator"));  writer.write("Sky condition: " + skyCondition);  writer.write(*System*.*getProperty*("line.separator"));  writer.write("Chance of rain: " + chanceOfRain + "%");  writer.write(*System*.*getProperty*("line.separator"));  writer.flush();  } catch (*FileNotFoundException exc*) {  *System*.out.println("FileNotFoundException" + *exc*.getMessage());  } catch (*IOException exc*) {  *System*.out.println("IOException: " + *exc*.getMessage());  } catch (*Exception exc*) {  *System*.out.println("Exception: " + *exc*.getMessage());  }  } } |

## ForecastTest.java

|  |
| --- |
| package *com.valencia*;  import *org.junit.jupiter.api.Assertions*; import *org.junit.jupiter.api.BeforeEach*; import *org.junit.jupiter.api.Test*;  import *java.io.ByteArrayOutputStream*; import *java.io.IOException*; import *java.io.*OutputStream; import *java.io.PrintStream*; import *java.nio.charset.StandardCharsets*; import *java.nio.file.Files*; import *java.nio.file.Path*; import *java.nio.file.Paths*;  public class *ForecastTest* {  final double DELTA = 0.1;  *Forecast* forecast;   *@BeforeEach* void setUp() {  forecast = new Forecast();  }   *@Test* void Forecast() {  *// Constructor without arguments  Forecast* defaultForecast = new Forecast();  *Assertions*.*assertEquals*(72, defaultForecast.getTemperature(), DELTA);  *Assertions*.*assertEquals*("clear", defaultForecast.getSkyCondition());  *Assertions*.*assertEquals*(0, defaultForecast.getChanceOfRain());   *// Constructor with custom arguments  Forecast* customForecast = new Forecast(149, "hurricane", 77);  *Assertions*.*assertEquals*(149, customForecast.getTemperature(), DELTA);  *Assertions*.*assertEquals*("hurricane", customForecast.getSkyCondition());  *Assertions*.*assertEquals*(77, customForecast.getChanceOfRain());  }   *@Test* void getTemperature() {  double actual = forecast.getTemperature();  double expected = 72;  *Assertions*.*assertEquals*(expected, actual, DELTA);  }   *@Test* void setTemperature() {  *// No value* forecast.setTemperature();  *Assertions*.*assertEquals*(72, forecast.getTemperature());   *// Invalid values* forecast.setTemperature(-101);  *Assertions*.*assertEquals*(72, forecast.getTemperature());  forecast.setTemperature(151);  *Assertions*.*assertEquals*(72, forecast.getTemperature());   *// Valid value* forecast.setTemperature(100);  *Assertions*.*assertEquals*(100, forecast.getTemperature());  }   *@Test* void getSkyCondition() {  *String* actual = forecast.getSkyCondition();  *String* expected = "clear";  *Assertions*.*assertEquals*(expected, actual);  }   *@Test* void setSkyCondition() {  *// No value* forecast.setSkyCondition();  *Assertions*.*assertEquals*("clear", forecast.getSkyCondition());   *// Valid values* forecast.setSkyCondition("raining");  *Assertions*.*assertEquals*("raining", forecast.getSkyCondition());  forecast.setSkyCondition("thunderstorm");  *Assertions*.*assertEquals*("thunderstorm", forecast.getSkyCondition());  }   *@Test* void getChanceOfRain() {  double actual = forecast.getChanceOfRain();  double expected = 0;  *Assertions*.*assertEquals*(expected, actual, DELTA);  }   *@Test* void setChanceOfRain() {  *// No value* forecast.setChanceOfRain();  *Assertions*.*assertEquals*(0, forecast.getChanceOfRain());   *// Invalid values* forecast.setChanceOfRain(-1);  *Assertions*.*assertEquals*(0, forecast.getChanceOfRain());  forecast.setTemperature(101);  *Assertions*.*assertEquals*(0, forecast.getChanceOfRain());   *// Valid values* forecast.setTemperature(99);  *Assertions*.*assertEquals*(99, forecast.getTemperature());  }   *@org.junit.jupiter.api.Test* void convertFahrenheitToCelsius() {  double actual = *Forecast*.*convertFahrenheitToCelsius*(100);  double expected = 37.78;  *Assertions*.*assertEquals*(expected, actual, DELTA);  }   *@org.junit.jupiter.api.Test* void convertCelsiusToFahrenheit() {  double actual = *Forecast*.*convertCelsiusToFahrenheit*(37.78);  double expected = 100;  *Assertions*.*assertEquals*(expected, actual, DELTA);  }   *@org.junit.jupiter.api.Test* void convertFahrenheitToKelvin() {  double actual = *Forecast*.*convertFahrenheitToKelvin*(32);  double expected = 273;  *Assertions*.*assertEquals*(expected, actual, DELTA);  }   *@org.junit.jupiter.api.Test* void convertKelvinToFahrenheit() {  double actual = *Forecast*.*convertKelvinToFahrenheit*(273);  double expected = 32;  *Assertions*.*assertEquals*(expected, actual, DELTA);  }   *@Test* void isRaining() {  *Assertions*.*assertFalse*(forecast.isRaining());  forecast.setSkyCondition("raining");  *Assertions*.*assertTrue*(forecast.isRaining());  }   *@Test* void print() {  OutputStream os = new ByteArrayOutputStream();  *PrintStream* printStream = new PrintStream(os);  *System*.*setOut*(printStream);   forecast.print();   *String* expected = "Temperature: 72.00 °F\nSky condition: clear\nChance of rain: 0%\n";  *Assertions*.*assertEquals*(expected, os.toString());  }   *@Test* void printToFile() {  *String* RESULT\_FILE = "forecast\_result.txt";  forecast.printToFile();   try {  *Path* path = *Paths*.*get*(RESULT\_FILE);  *String* actual = *Files*.*readString*(path, *StandardCharsets*.UTF\_8);  *String* expected = "Temperature: 72.00 °F\nSky condition: clear\nChance of rain: 0%\n";  *Assertions*.*assertEquals*(expected, actual);  *Files*.*deleteIfExists*(*Path*.*of*(RESULT\_FILE));  } catch (*IOException exc*) {  *Assertions*.*fail*(*exc*.getMessage());  }  } } |

## Test

Text

Description automatically generated

# Part 2

## Forecast.java

|  |
| --- |
| public void printToFile() {  try (*FileWriter* writer = new FileWriter(RESULT\_FILE)) {  writer.write(*String*.*format*("Temperature: %.2f °F", temperature));  writer.write(*System*.*getProperty*("line.separator"));  writer.write("Sky condition: " + skyCondition);  writer.write(*System*.*getProperty*("line.separator"));  writer.write("Chance of rain: " + chanceOfRain + "%");  writer.write(*System*.*getProperty*("line.separator"));  writer.flush();  } catch (*FileNotFoundException exc*) {  *System*.out.println("FileNotFoundException" + *exc*.getMessage());  } catch (*IOException exc*) {  *System*.out.println("IOException: " + *exc*.getMessage());  } catch (*Exception exc*) {  *System*.out.println("Exception: " + *exc*.getMessage());  } } |

## ForecastTest.java

|  |
| --- |
| *@Test* void printToFile() {  *String* RESULT\_FILE = "forecast\_result.txt";  *Forecast* forecast = new Forecast();  forecast.printToFile();   try {  *Path* path = *Paths*.*get*(RESULT\_FILE);  *String* actual = *Files*.*readString*(path, *StandardCharsets*.UTF\_8);  *String* expected = "Temperature: 72.00 °F\nSky condition: clear\nChance of rain: 0%\n";  *Assertions*.*assertEquals*(expected, actual);  *Files*.*deleteIfExists*(*Path*.*of*(RESULT\_FILE));  } catch (*IOException exc*) {  *Assertions*.*fail*(*exc*.getMessage());  } } |

## Test

# Text Description automatically generated

# Part 3