**(TEMPERATURE.java)**

**package** forecastMe;

**import** java.util.Scanner;

**public** **class** Temperature {

**private** String sky = "clear";

**private** **double** temperatureOutside = 0.0;

**private** **int** chance = 0;

**public** **void** setSkies(**int** userChanceOfRain) {

chance = userChanceOfRain;

}

**public** **void** setCondition(String userSkyCondition) {

sky = userSkyCondition;

}

**public** **void** setTemperature(**double** userTemperature) {

temperatureOutside = userTemperature;

}

**public** **void** print() {

System.***out***.println(sky + " -- " + chance + "%%");

System.***out***.printf("%.1f", temperatureOutside);

}

}

(forecastMe.java)

**package** forecastMe;

**import** java.util.\*;

**public** **class** forecastMe {

**public** **static** **void** main(String[] args) {

Temperature forecast = **new** Temperature();

**double** userTemp = 0;

**int** select = 0;

**boolean** weather;

System.***out***.println("What temperature would you like to convert this to?");

System.***out***.println("1. Fahrenheit to Celsius");

System.***out***.println("2. Celsius to Fahrenheit");

System.***out***.println("3. Fahrenheit to Kelvin");

System.***out***.println("4. Kelvin to Fahrenheit");

Scanner user\_input = **new** Scanner(System.***in***);

select = user\_input.nextInt();

**switch**(select) {

**case** 1: // Fahrenheit to Celsius

userTemp = userTemp - 273.15;

**break**;

**case** 2: // Celsius to Fahrenheit

userTemp = (5.0/9.0 \* userTemp) + 32.0;

**break**;

**case** 3: // Fahrenheit to Kelvin

userTemp = (userTemp - 32.0) \* 5.0/9.0 + 273.15;

**break**;

**case** 4: // Kelvin to Fahrenheit

userTemp = (userTemp - 273.15) \* 9.0/5.0 + 32.0;

**break**;

**default**:

System.***out***.println("Try again");

}

System.***out***.println("Will it rain today?");

forecast.print();

forecast.setSkies(80); // chance

forecast.setTemperature(userTemp); // temperature

forecast.setCondition("cloudy"); // condition of skies

forecast.print();

}

}

**Part 3:**

An **object** is a grouping of data (variables) and operations that can be performed on that data, while the **class** construct defines a new type that can group data and methods to form an object. There are moments when a programmer uses abstraction to have a user interact with an item at a high-level, with lower-level internal details hidden from the user (aka information hiding or encapsulation).

With the use of classes, a class user doesn’t need to know the data and methods are implemented. The user will only need to understand how each public member method behaves and what kind of values are provided by those functions. At that moment, a programmer may use the **new** operator, to allocate an object of the specified class type.

