



UNIVERSITI
TEKNOLOGI
PETRONAS

TEB1043 Object Oriented Programming

Department of Computer and Information Sciences

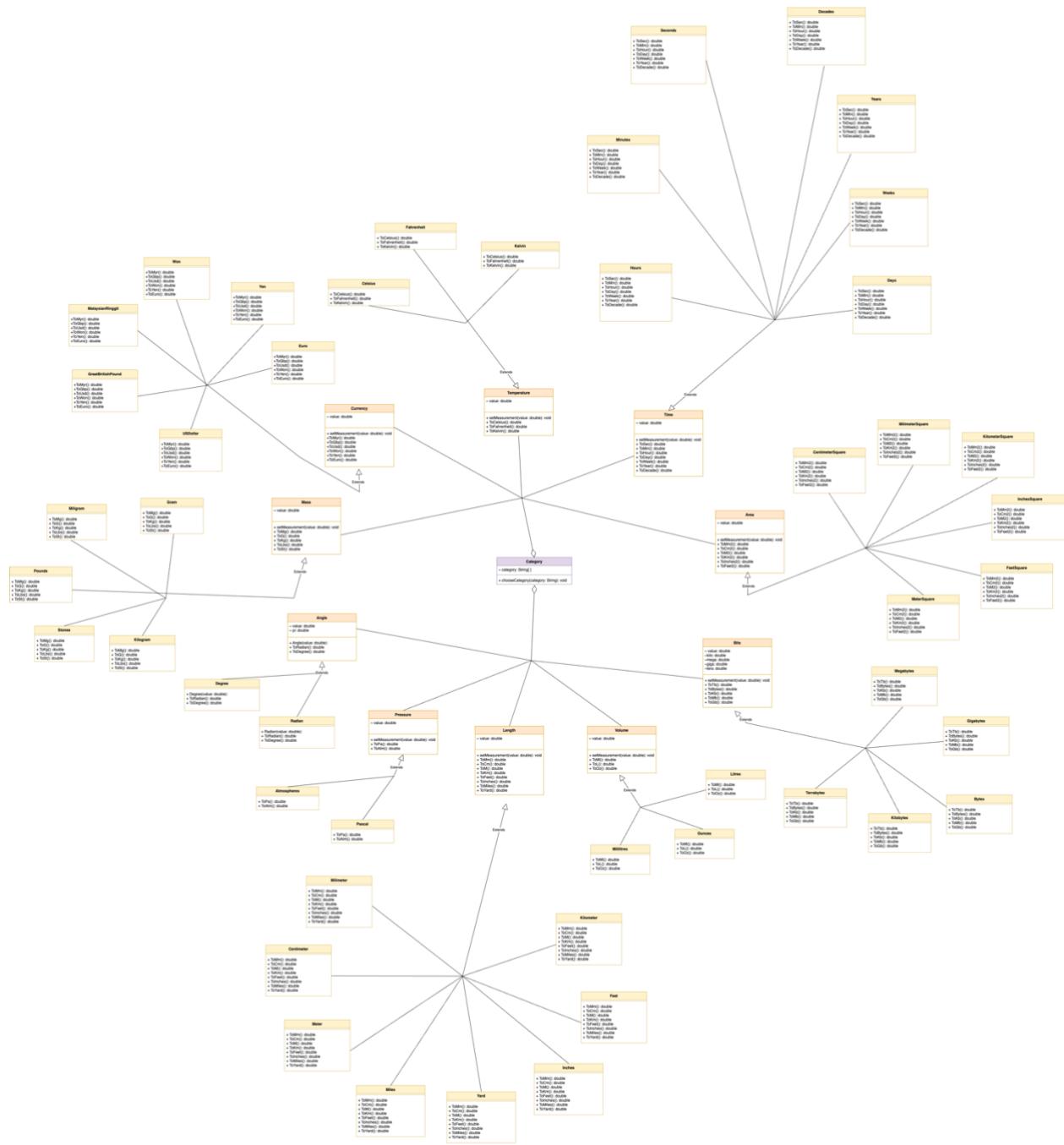
Prepared for:

Dr Mohamed Nordin bin Zakaria

Prepared by:

Name	Student ID	Program
Ahmad Zafri bin Mohd Ramli	21000556	Bachelor of Computer Science
Iman Kamil bin Suhaimi	21000175	Bachelor of Computer Science
Ammar Afif bin Mohd Khairie	21001232	Bachelor of Computer Science
Muhammad Aqil Fauzan bin Muhammad Hafidzi	21001113	Bachelor of Computer Science
Mohamad Izhar bin Ahmad Nordin	21001242	Bachelor of Computer Science
Danish Hariz bin Yusrizal	21001071	Bachelor of Computer Science
Imran Haqeem bin Mohd Khassan	21001366	Bachelor of Computer Science
Ikhwan Hanafi bin Irwan	21001245	Bachelor of Computer Science
Aiman Shuhailal bin Shuhaimi	21001293	Bachelor of Computer Science
Muhammad Faris bin Mohd Rusli	21001346	Bachelor of Computer Science

Project UML



Members of Each Class

Class	Assignee
Length	Imran Haqeem
Mass	Muhammad Faris
Time	Aiman Shuhail
Temperature	Mohamad Izhar
Currency	Iman Kamil
Angle Category	Ammar Afif
Pressure	Ikhwan Hanafi
Area	Danish Hariz
Volume	Ahmad Zafri
Bits	Aqil Fauzan

Angle

Main Function (Test Code for Angle)

```
16 ►  public static void main(String[] args) {
17     Scanner in = new Scanner(System.in);
18     Angle angle;
19
20     System.out.println("\n---Choose angle unit to convert from---\n\tDegree (D)\n\tRadian (R)");
21     //get unit to convert from
22     System.out.print("Choose unit to convert FROM (D/R): ");
23     String unitFrom = in.nextLine().toUpperCase();
24     while (!unitFrom.equalsIgnoreCase("D") && !unitFrom.equalsIgnoreCase("R")){
25         System.out.println("Invalid unit!");
26         System.out.print("Choose unit to convert FROM (D/R): ");
27         unitFrom = in.nextLine().toUpperCase();
28     }
29     //get unit to convert to
30     System.out.print("Choose unit to convert TO (D/R): ");
31     String unitTo = in.nextLine().toUpperCase();
32     while (!unitTo.equalsIgnoreCase("D") && !unitTo.equalsIgnoreCase("R")){
33         System.out.println("Invalid unit!");
34         System.out.print("Choose unit to convert TO (D/R): ");
35         unitTo = in.nextLine().toUpperCase();
36     }
37     //get value input from user
38     System.out.print("\nEnter value of angle: ");
39     double value = in.nextDouble();
40
41     if (unitFrom.equalsIgnoreCase("D")){
42         angle = new Degree(value);
43         switch (unitTo) {
44             case "D" ->
45                 System.out.println("\n" + value + " " + "degree in degree: " + String.format("%.4f", angle.toDegree()));
46             case "R" ->
47                 System.out.println("\n" + value + " " + "degree in radian: " + String.format("%.4f", angle.toRadian()));
48         }
49     } else {
50         angle = new Radian(value);
51         switch (unitTo) {
52             case "D" ->
53                 System.out.println("\n" + value + " " + "radian in degree: " + String.format("%.4f", angle.toDegree()));
54             case "R" ->
55                 System.out.println("\n" + value + " " + "radian in radian: " + String.format("%.4f", angle.toRadian()));
56         }
57     }
58     in.close();
59 }
60 }
```

Angle Class

```
2 inheritors
3  ⚡ public abstract class Angle {
4
5      //attributes
6      protected double value;
7      protected final double pi = Math.PI;
8
9      //constructor
10     public Angle(double value) { this.value = value; }
11
12
13
14      //abstract methods headers
15      2 implementations
15  ⚡     public abstract double toDegree();
16      2 implementations
16  ⚡     public abstract double toRadian();
17
18      //create subclass of Angle
19      ⚡ class Degree extends Angle{
20          //constructor that creates Degree object and assign the value measurement
21          public Degree(double value) { super(value); }
22
23
24          //abstract methods with calculations (body)
25          ⚡     public double toDegree() { return value; }
26          ⚡     public double toRadian() { return (value * pi / 180); }
27
28      ⚡ }
29
30
31      //create subclass of Angle
32      ⚡ class Radian extends Angle{
33          //constructor that creates Degree object and assign the value measurement
34          public Radian(double value) { super(value); }
35
36
37          //abstract methods with calculations (body)
38          ⚡     public double toDegree() { return(value * 180 / pi); }
39          ⚡     public double toRadian() { return value; }
40
41      ⚡ }
```

Temperature

```
J TempConverter.java X J Temperature.java
src > J TempConverter.java > TempConverter > main(String[])
1 import java.text.DecimalFormat;
2 import java.text.NumberFormat;
3 import java.util.*;
4
5 public class TempConverter{
6     Run | Debug
7     public static void main(String[] args) throws Exception {
8         Scanner input = new Scanner(System.in);
9         NumberFormat formatter = new DecimalFormat(pattern: "#0.0");
10
11         System.out.print(s: "Enter temperature value: ");
12         double tempValue = input.nextDouble();
13
14         System.out.print(s: "Enter temperature unit (C/F/K): ");
15         String fromUnit = input.next();
16
17         System.out.print(s: "Enter unit to convert to (C/F/K): ");
18         String toUnit = input.next();
19         input.close();
20
21
22         // Creates an instance of the class Temperature
23         Temperature initialTemp;
24         if (fromUnit.equalsIgnoreCase(anotherString: "C")){
25             initialTemp = new Celcius(tempValue); // Instantiates a Celcius object if user entered "C"
26         }else if (fromUnit.equalsIgnoreCase(anotherString: "F")){
27             initialTemp = new Farenheit(tempValue); // Instantiates a Farenheit object if user entered "F"
28         }else if (fromUnit.equalsIgnoreCase(anotherString: "K")){
29             initialTemp = new Kelvin(tempValue); // Instantiates a Kelvin object if user entered "K"
30         }else{
31             System.out.print(s: "Invalid temperature unit!");
32             return;
33         }
34     }
```

```
J TempConverter.java × J Temperature.java
src > J TempConverter.java > TempConverter > main(String[])
  36 // Declare a new variable to store converted unit
  37     double convertedTemp;
  38     if (toUnit.equalsIgnoreCase(anotherString: "C")){
  39         convertedTemp = initialTemp.toCelcius();
  40         System.out.println("Temperature in Celcius: " + formatter.format(convertedTemp));
  41     }else if (toUnit.equalsIgnoreCase(anotherString: "F")){
  42         convertedTemp = initialTemp.toFahrenheit();
  43         System.out.println("Temperature in Fahrenheit: " + formatter.format(convertedTemp));
  44     }else if (toUnit.equalsIgnoreCase(anotherString: "K")){
  45         convertedTemp = initialTemp.toKelvin();
  46         System.out.println("Temperature in Kelvin: " + formatter.format(convertedTemp));
  47     }else{
  48         System.out.print(s: "Invalid temperature unit!");
  49         return;
  50     }
  51 }
  52 }
  53 }
  54
  55 // Inherited classes
  56 class Celcius extends Temperature{
  57     public Celcius(double value){
  58         super(value);
  59     }
  60
  61     public double toCelcius(){
  62         return value;
  63     }
  64
  65     public double toFahrenheit(){
  66         return (value * 9 / 5) + 32;
  67     }
  68
  69     public double toKelvin(){
  70         return (value + 273.15);
  71     }
  72 }
  73 }
```

```
J TempConverter.java X J Temperature.java
src > J TempConverter.java > TempConverter > main(String[])
73 }
74
75 class Fahrenheit extends Temperature{
76     public Fahrenheit(double value){
77         super(value);
78     }
79
80     public double toCelcius(){
81         return (value - 32) * 5 / 9;
82     }
83
84     public double toFahrenheit(){
85         return value;
86     }
87
88     public double toKelvin(){
89         return toCelcius() + 273.15;
90     }
91 }
92
93 class Kelvin extends Temperature{
94     public Kelvin(double value){
95         super(value);
96     }
97
98     public double toCelcius(){
99         return (value - 273.15);
100    }
101
102    public double toFahrenheit(){
103        return (value - 273.15) * 9 / 5 + 32;
104    }
105
106    public double toKelvin(){
107        return value;
108    }
109 }
```

J TempConverter.java J Temperature.java X

```
src > J Temperature.java > ⌂ Temperature > ⌂ toCelcius()
1 abstract class Temperature {
2     protected double value;
3
4     public Temperature(double value){
5         this.value = value;
6     }
7
8     public abstract double toCelcius();
9
10    public abstract double toFahrenheit();
11
12    public abstract double toKelvin();
13
14 }
15
```

Volume

Main class

```
1 import ...
2 public class Main {
3     public static void main(String[] args) throws Exception{
4         Scanner input = new Scanner(System.in);
5         NumberFormat formatter = new DecimalFormat(pattern: "#0.0");
6
6         System.out.println("1. (L) Litre \n2. (M) Milliliter \n3. (F) Fluid Ounce\n");
7
8         System.out.print("Select unit to convert from (L/M/F): ");
9         String unit = input.next().toUpperCase();
10
11         System.out.print("Enter volume value: ");
12         double volumeValue = input.nextDouble();
13
14         System.out.print("\nEnter unit to convert to (L/M/F): ");
15         String unit2 = input.next().toUpperCase();
16         input.close();
17
18         Volume volume;
19         if (unit.equalsIgnoreCase("L")){
20             volume = new Litre(volumeValue);
21             switch (unit2){
22                 case "L":
23                     System.out.println("\n" + volumeValue + " Litre in Litre: " + String.format("%.2f", volume.toLitre()));
24                     break;
25                 case "M":
26                     System.out.println("\n" + volumeValue + " Litre in Milliliter: " + String.format("%.2f", volume.toMilliliter()));
27                     break;
28                 case "F":
29                     System.out.println("\n" + volumeValue + " Litre in Fluid Ounce: " + String.format("%.2f", volume.toOunces()));
30                     break;
31             }
32         }else if (unit.equalsIgnoreCase("M")){
33             volume = new Milliliter(volumeValue);
34             switch (unit2){
35                 case "L":
36                     System.out.println("\n" + volumeValue + " Milliliter in Litre: " + String.format("%.2f", volume.toLitre()));
37                     break;
38                 case "M":
39                     System.out.println("\n" + volumeValue + " Milliliter in Milliliter: " + String.format("%.2f", volume.toMilliliter()));
40                     break;
41                 case "F":
42                     System.out.println("\n" + volumeValue + " Milliliter in Fluid Ounce: " + String.format("%.2f", volume.toOunces()));
43                     break;
44             }
45         }else if (unit.equalsIgnoreCase("F")){
46             volume = new Dounces(volumeValue);
47             switch (unit2){
48                 case "L":
49                     System.out.println("\n" + volumeValue + " Fluid Ounces in Litre: " + String.format("%.2f", volume.toLitre()));
50                     break;
51                 case "M":
52                     System.out.println("\n" + volumeValue + " Fluid Ounces in Milliliter: " + String.format("%.2f", volume.toMilliliter()));
53                     break;
54                 case "F":
55                     System.out.println("\n" + volumeValue + " Fluid Ounces in Fluid Ounce: " + String.format("%.2f", volume.toOunces()));
56                     break;
57             }
58         }else{
59             System.out.print("Invalid currency unit");
60             return;
61         }
62     }
63 }
64 
```

Volume Class

The screenshot shows a Java code editor interface with two tabs: "Main.java" and "Volume.java". The "Volume.java" tab is active, displaying the following code:

```
1  abstract class Volume {  
2      protected double value;  
3  
4      public Volume(double value) { this.value = value; }  
5  
6      public abstract double toLitre();  
7  
8      public abstract double toOunces();  
9  
10     public abstract double toMilliliter();  
11  
12 }  
13  
14 class Litre extends Volume{  
15     public Litre(double value) { super(value); }  
16     public double toLitre() { return value; }  
17     public double toOunces(){  
18         return value * 33.81;  
19     }  
20     public double toMilliliter() { return value * 1000; }  
21 }  
22  
23 class Ounces extends Volume{  
24     public Ounces(double value) { super(value); }  
25     public double toLitre(){  
26         return value / 33.814;  
27     }  
28     public double toOunces(){  
29         return value;  
30     }  
31     public double toMilliliter() { return value * 29.574; }  
32 }  
33  
34 class Milliliter extends Volume{  
35     public Milliliter(double value) { super(value); }  
36     public double toLitre(){  
37         return value / 1000;  
38     }  
39     public double toOunces(){  
40         return value / 29.57;  
41     }  
42     public double toMilliliter() { return value; }  
43 }  
44  
45 }
```

The code defines an abstract class `Volume` with three abstract methods: `toLitre()`, `toOunces()`, and `toMilliliter()`. It then defines three concrete classes: `Litre`, `Ounces`, and `Milliliter`, each extending `Volume` and implementing these methods. The `Litre` class has a constructor taking a double value and returning its value. The `Ounces` class has a constructor taking a double value and returning its value. The `Milliliter` class has a constructor taking a double value and returning its value.

Length

Main Class

```
1 import java.text.DecimalFormat;
2 import java.text.NumberFormat;
3 import java.util.*;
4
5 no usages
6 ► public class Main {
7     no usages
8     ► public static void main(String[] args) throws Exception{
9         Scanner input = new Scanner(System.in);
10        NumberFormat formatter = new DecimalFormat( pattern: "#0.0");
11
12        System.out.print("Enter length value: ");
13        double lengthValue = input.nextDouble();
14
15        System.out.print("Enter length unit (MM/CM/MI/YD/IN/FT/KM): ");
16        String unit = input.next().toUpperCase();
17
18        System.out.print("Select new length unit (MM/CM/MI/YD/IN/FT/KM): ");
19        String newUnit = input.next().toUpperCase();
20        input.close();
21
22        Length length;
23        if (unit.equalsIgnoreCase( anotherString: "MM")){
24            length = new Millimeter(lengthValue);
25            switch (newUnit){
26                case "MM":
27                    System.out.println("\n" + lengthValue + " MM in MM: " + String.format("%.4f",length.toMillimeter()));
28                    break;
29                case "CM":
30                    System.out.println("\n" + lengthValue + " MM in CM: " + String.format("%.4f",length.toCentimeter()));
31                    break;
32                case "M":
33                    System.out.println("\n" + lengthValue + " MM in M: " + String.format("%.4f",length.toMeter()));
34                    break;
35                case "MI":
36                    System.out.println("\n" + lengthValue + " MM in MI: " + String.format("%.4f",length.toMiles()));
37                    break;
38                case "YD":
39                    System.out.println("\n" + lengthValue + " MM in YD: " + String.format("%.4f",length.toYard()));
40                    break;
41                case "IN":
42                    System.out.println("\n" + lengthValue + " MM in IN: " + String.format("%.4f",length.toInches()));
43                    break;
44                case "FT":
45                    System.out.println("\n" + lengthValue + " MM in FT: " + String.format("%.4f",length.toFeet()));
46                    break;
47                case "KM":
48                    System.out.println("\n" + lengthValue + " MM in KM: " + String.format("%.4f",length.toKilometer()));
49                    break;
50            }
51        }else if (unit.equalsIgnoreCase( anotherString: "CM")){
52            length = new Centimeter(lengthValue);
53            switch (newUnit){
54                case "MM":
55                    System.out.println("\n" + lengthValue + " CM in MM: " + String.format("%.4f",length.toMillimeter()));
56                    break;
57                case "CM":
58                    System.out.println("\n" + lengthValue + " CM in CM: " + String.format("%.4f",length.toCentimeter()));
59                    break;
60                case "M":
61                    System.out.println("\n" + lengthValue + " CM in M: " + String.format("%.4f",length.toMeter()));
62                    break;
63            }
64        }
65    }
66}
```

```

61         case "MI":
62             System.out.println("\n" + lengthValue + " CM in MI: " + String.format("%.4f",length.toMiles()));
63             break;
64         case "YD":
65             System.out.println("\n" + lengthValue + " CM in YD: " + String.format("%.4f",length.toYard()));
66             break;
67         case "IN":
68             System.out.println("\n" + lengthValue + " CM in IN: " + String.format("%.4f",length.toInches()));
69             break;
70         case "FT":
71             System.out.println("\n" + lengthValue + " CM in FT: " + String.format("%.4f",length.toFeet()));
72             break;
73         case "KM":
74             System.out.println("\n" + lengthValue + " CM in KM: " + String.format("%.4f",length.toKilometer()));
75             break;
76     }
77 }else if (unit.equalsIgnoreCase(anotherString: "M")){
78     length = new Meter(lengthValue);
79     switch (newUnit){
80         case "MM":
81             System.out.println("\n" + lengthValue + " M in MM: " + String.format("%.4f",length.toMilimeter()));
82             break;
83         case "CM":
84             System.out.println("\n" + lengthValue + " M in CM: " + String.format("%.4f",length.toCentimeter()));
85             break;
86         case "M":
87             System.out.println("\n" + lengthValue + " M in M: " + String.format("%.4f",length.toMeter()));
88             break;
89         case "IN":
90             System.out.println("\n" + lengthValue + " M in MI: " + String.format("%.4f",length.toMiles()));
91             break;
92         case "YD":
93             System.out.println("\n" + lengthValue + " M in YD: " + String.format("%.4f",length.toYard()));
94             break;
95         case "IN":
96             System.out.println("\n" + lengthValue + " M in IN: " + String.format("%.4f",length.toInches()));
97             break;
98         case "FT":
99             System.out.println("\n" + lengthValue + " M in FT: " + String.format("%.4f",length.toFeet()));
100            break;
101        case "KM":
102            System.out.println("\n" + lengthValue + " M in KM: " + String.format("%.4f",length.toKilometer()));
103            break;
104    }
105 }else if (unit.equalsIgnoreCase(anotherString: "MI")){
106     length = new Miles(lengthValue);
107     switch (newUnit){
108         case "MM":
109             System.out.println("\n" + lengthValue + " MI in MM: " + String.format("%.4f",length.toMilimeter()));
110             break;
111         case "CM":
112             System.out.println("\n" + lengthValue + " MI in CM: " + String.format("%.4f",length.toCentimeter()));
113             break;
114         case "M":
115             System.out.println("\n" + lengthValue + " MI in M: " + String.format("%.4f",length.toMeter()));
116             break;
117         case "IN":
118             System.out.println("\n" + lengthValue + " MI in MI: " + String.format("%.4f",length.toMiles()));
119             break;
120         case "YD":
121             System.out.println("\n" + lengthValue + " MI in YD: " + String.format("%.4f",length.toYard()));
122             break;
123     }
124 }
```

```

125         case "IN":
126             System.out.println("\n" + lengthValue + " MI in IN: " + String.format("%.4f",length.toInches()));
127             break;
128         case "FT":
129             System.out.println("\n" + lengthValue + " MI in FT: " + String.format("%.4f",length.toFeet()));
130             break;
131         case "KM":
132             System.out.println("\n" + lengthValue + " MI in KM: " + String.format("%.4f",length.toKilometer()));
133             break;
134     }
135     else if (unit.equalsIgnoreCase(anotherString: "YD")){
136         length = new Yard(lengthValue);
137         switch (newUnit){
138             case "MM":
139                 System.out.println("\n" + lengthValue + " YD in MM: " + String.format("%.4f",length.toMilimeter()));
140                 break;
141             case "CM":
142                 System.out.println("\n" + lengthValue + " YD in CM: " + String.format("%.4f",length.toCentimeter()));
143                 break;
144             case "M":
145                 System.out.println("\n" + lengthValue + " YD in M: " + String.format("%.4f",length.toMeter()));
146                 break;
147             case "MI":
148                 System.out.println("\n" + lengthValue + " YD in MI: " + String.format("%.4f",length.toMiles()));
149                 break;
150             case "YD":
151                 System.out.println("\n" + lengthValue + " YD in YD: " + String.format("%.4f",length.toYard()));
152                 break;
153             case "IN":
154                 System.out.println("\n" + lengthValue + " YD in IN: " + String.format("%.4f",length.toInches()));
155                 break;
156             case "FT":
157                 System.out.println("\n" + lengthValue + " YD in FT: " + String.format("%.4f",length.toFeet()));
158                 break;
159             case "KM":
160                 System.out.println("\n" + lengthValue + " YD in KM: " + String.format("%.4f",length.toKilometer()));
161                 break;
162         }
163     }
164     else if (unit.equalsIgnoreCase(anotherString: "IN")){
165         length = new Inches(lengthValue);
166         switch (newUnit){
167             case "MM":
168                 System.out.println("\n" + lengthValue + " IN in MM: " + String.format("%.4f",length.toMilimeter()));
169                 break;
170             case "CM":
171                 System.out.println("\n" + lengthValue + " IN in CM: " + String.format("%.4f",length.toCentimeter()));
172                 break;
173             case "M":
174                 System.out.println("\n" + lengthValue + " IN in M: " + String.format("%.4f",length.toMeter()));
175                 break;
176             case "MI":
177                 System.out.println("\n" + lengthValue + " IN in MI: " + String.format("%.4f",length.toMiles()));
178                 break;
179             case "YD":
180                 System.out.println("\n" + lengthValue + " IN in YD: " + String.format("%.4f",length.toYard()));
181                 break;
182             case "IN":
183                 System.out.println("\n" + lengthValue + " IN in IN: " + String.format("%.4f",length.toInches()));
184                 break;
185             case "FT":
186                 System.out.println("\n" + lengthValue + " IN in FT: " + String.format("%.4f",length.toFeet()));
187                 break;
188         }
189     }

```

```

185         case "KM":
186             System.out.println("\n" + lengthValue + " IN in KM: " + String.format("%.4f",length.toKilometer()));
187             break;
188     }
189     else if (unit.equalsIgnoreCase("FT")){
190         length = new Feet(lengthValue);
191         switch (newUnit){
192             case "MM":
193                 System.out.println("\n" + lengthValue + " FT in MM: " + String.format("%.4f",length.toMilimeter()));
194                 break;
195             case "CM":
196                 System.out.println("\n" + lengthValue + " FT in CM: " + String.format("%.4f",length.toCentimeter()));
197                 break;
198             case "M":
199                 System.out.println("\n" + lengthValue + " FT in M: " + String.format("%.4f",length.toMeter()));
200                 break;
201             case "MI":
202                 System.out.println("\n" + lengthValue + " FT in MI: " + String.format("%.4f",length.toMiles()));
203                 break;
204             case "YD":
205                 System.out.println("\n" + lengthValue + " FT in YD: " + String.format("%.4f",length.toYard()));
206                 break;
207             case "IN":
208                 System.out.println("\n" + lengthValue + " FT in IN: " + String.format("%.4f",length.toInches()));
209                 break;
210             case "FT":
211                 System.out.println("\n" + lengthValue + " FT in FT: " + String.format("%.4f",length.toFeet()));
212                 break;
213             case "KM":
214                 System.out.println("\n" + lengthValue + " FT in KM: " + String.format("%.4f",length.toKilometer()));
215                 break;
216         }
217     else if (unit.equalsIgnoreCase("KM")){
218         length = new Kilometer(lengthValue);
219         switch (newUnit){
220             case "MM":
221                 System.out.println("\n" + lengthValue + " KM in MM: " + String.format("%.4f",length.toMilimeter()));
222                 break;
223             case "CM":
224                 System.out.println("\n" + lengthValue + " KM in CM: " + String.format("%.4f",length.toCentimeter()));
225                 break;
226             case "M":
227                 System.out.println("\n" + lengthValue + " KM in M: " + String.format("%.4f",length.toMeter()));
228                 break;
229             case "MI":
230                 System.out.println("\n" + lengthValue + " KM in MI: " + String.format("%.4f",length.toMiles()));
231                 break;
232             case "YD":
233                 System.out.println("\n" + lengthValue + " KM in YD: " + String.format("%.4f",length.toYard()));
234                 break;
235             case "IN":
236                 System.out.println("\n" + lengthValue + " KM in IN: " + String.format("%.4f",length.toInches()));
237                 break;
238             case "FT":
239                 System.out.println("\n" + lengthValue + " KM in FT: " + String.format("%.4f",length.toFeet()));
240                 break;
241             case "KM":
242                 System.out.println("\n" + lengthValue + " KM in KM: " + String.format("%.4f",length.toKilometer()));
243                 break;
244         }
245     }else{
246         System.out.print("Invalid length unit");
247         return;

```

Length Class

```
1 ⏷ abstract class Length {  
2     65 usages  
3         protected double value;  
4  
5             8 usages  
6     6 ⏷ public Length(double value) {this.value = value;}  
7  
8             8 usages 8 implementations  
9     7 ⏷ public abstract double toMilimeter();  
10    8 usages 8 implementations  
11    7 ⏷ public abstract double toCentimeter();  
12        8 usages 8 implementations  
13    8 ⏷ public abstract double toMeter();  
14        8 usages 8 implementations  
15    9 ⏷ public abstract double toMiles();  
16        8 usages 8 implementations  
17    10 ⏷ public abstract double toYard();  
18        8 usages 8 implementations  
19    11 ⏷ public abstract double toInches();  
20        8 usages 8 implementations  
21    12 ⏷ public abstract double toFeet();  
22        8 usages 8 implementations  
23    13 ⏷ public abstract double toKilometer();  
24  
25        no usages  
26    14 ⏷ class Milimeter extends Length{  
27        no usages  
28    15 ⏷     public Milimeter(double value) { super(value); }  
29        8 usages  
30    16 ⏷         public double toMilimeter(){return value;}  
31
```

```
20 ⚡    public double toCentimeter() { return value / 10.0; }
      8 usages
23 ⚡    public double toMeter() {return value / 1000.0;}
24
      8 usages
25 ⚡    public double toMiles() { return value / 1609344.0; }
      8 usages
28 ⚡    public double toYard() { return value / 914.4; }
      8 usages
31 ⚡    public double toInches() { return value / 25.4; }
      8 usages
34 ⚡    public double toFeet(){return value / 304.8;}
      8 usages
35 ⚡    public double toKilometer() { return value / 1000000.0; }
38  }
39
      no usages
40 class Centimeter extends Length{
      no usages
41     public Centimeter(double value) { super(value); }
      8 usages
44 ⚡    public double toMilimeter() { return value * 10.0; }
      8 usages
47 ⚡    public double toCentimeter() { return value; }
50
      8 usages
51 ⚡    public double toMeter() {return value / 100.0;}
52
      8 usages
53 ⚡    public double toMiles(){return value / 160934.4;}
```

```
54 ⚡↑    public double toYard(){return value / 91.44;}
55 ⚡↑    8 usages
56 ⚡↑    public double toInches(){return value / 2.54;}
57 ⚡↑    8 usages
58 ⚡↑    public double toFeet() { return value / 30.48; }
59 ⚡↑    8 usages
60 ⚡↑    public double toKilometer() { return value / 100000.0; }
61 ⚡↑    }
62 ⚡↑    no usages
63
64 ⚡ class Meter extends Length {
65     no usages
66     public Meter(double value) {super(value);}
67     8 usages
68     public double toMilimeter() {return value * 1000.0;}
69 ⚡↑    8 usages
70     public double toCentimeter() {return value * 100.0;}
71 ⚡↑    8 usages
72     public double toMeter() {return value;}
73 ⚡↑    8 usages
74     public double toMiles() {return value / 1609.0;}
75 ⚡↑    8 usages
76     public double toYard() {return value * 1.094;}
```

```
    8 usages
77 ⬤    public double toInches() {return value * 39.37;}
78
79 ⬤    8 usages
80      public double toFeet() {return value * 3.281;}
81 ⬤    8 usages
82      public double toKilometer() {return value / 1000.0;}
83    }
84      no usages
85    class Miles extends Length {
86      no usages
87        public Miles(double value) { super(value); }
88        8 usages
89        public double toMilimeter() { return value * 160934; }
90        8 usages
91        public double toCentimeter() { return value * 160934.4; }
92
93        8 usages
94        public double toMeter() { return value * 1609.344; }
95
96        8 usages
97        public double toMiles() { return value; }
98
99        8 usages
100       public double toYard() {return value * 1760;}
101
102      8 usages
103      public double toInches() {return value * 63360; }
104
105
```

```
108 ⚡  public double toFeet() { return value * 5280; }
111
112 ⚡  8 usages
113  } no usages
114
115  class Yard extends Length {
116      no usages
117      public Yard(double value) { super(value); }
118
119      8 usages
120  ⚡  public double toMilimeter() { return value * 914.4; }
121
122      8 usages
123  ⚡  public double toCentimeter() { return value * 91.44; }
124
125      8 usages
126  ⚡  public double toMeter() {
127      return value * 0.9144;
128  }
129
130      8 usages
131  ⚡  public double toMiles() { return value / 1760; }
132
133      8 usages
134  ⚡  public double toYard() { return value; }
135
136      8 usages
137  ⚡  public double toInches() { return value * 36; }
138
139
140
141
142
143
144
```

```
145 ⚡  public double toFeet() { return value * 3; }
148
149 ⚡  8 usages
150 ⚡  public double toKilometer() { return value * 0.0009144; }
152 ⚡  }
153 ⚡  no usages
153 ⚡  class Inches extends Length {
154 ⚡  no usages
154 ⚡  public Inches(double value) { super(value); }
157
158 ⚡  8 usages
158 ⚡  public double toMilimeter() { return value * 25.4; }
161
162 ⚡  8 usages
162 ⚡  public double toCentimeter() { return value * 2.54; }
165
166 ⚡  8 usages
166 ⚡  public double toMeter() { return value * 0.0254; }
169
170 ⚡  8 usages
170 ⚡  public double toMiles() { return value / 63360; }
173
174 ⚡  8 usages
174 ⚡  public double toYard() { return value / 36; }
177
178 ⚡  8 usages
178 ⚡  public double toInches() { return value; }
181
182 ⚡  8 usages
182 ⚡  public double toFeet() { return value / 12; }
185
```

```
186 ⚡}     public double toKilometer() { return value * 0.0000254; }
189 }
190
191     no usages
192     class Feet extends Length {
193         no usages
194         public Feet(double value) { super(value); }
195
196         8 usages
197         public double toMilimeter() { return value * 304.8; }
198
199         8 usages
200         public double toCentimeter() { return value * 30.48; }
201
202         8 usages
203         public double toMeter() { return value * 0.3048; }
204
205         8 usages
206         public double toMiles() { return value / 5280; }
207
208         8 usages
209         public double toYard() { return value / 3; }
210
211         8 usages
212         public double toInches() { return value * 12; }
213
214         8 usages
215         public double toFeet() { return value; }
216
217         8 usages
218         public double toKilometer() { return value / 3280.84; }
219
220
221
222 ⚡}     public double toKilometer() { return value / 3280.84; }
```

```
227  class Kilometer extends Length {  
228      no usages  
231      8 usages  
232  ↗    public double toMillimeter() { return value * 1000000; }  
235      8 usages  
236  ↗    public double toCentimeter() { return value * 10000; }  
239      8 usages  
240  ↗    public double toMeter() { return value * 1000; }  
243      8 usages  
244  ↗    public double toMiles() { return value / 1.609344; }  
247      8 usages  
248  ↗    public double toYard() { return value * 1093.61; }  
251      8 usages  
252  ↗    public double toInches() { return value * 39370.1; }  
255      8 usages  
256  ↗    public double toFeet() { return value * 3280.84; }  
259      8 usages  
260  ↗    public double toKilometer() { return value; }  
263  ↘ }  
264
```

Mass

Main Class

```
1 import java.util.*;
2 no usages
3 ► public class Main {
4     no usages
5     ►     public static void main(String[] args) {
6         Scanner input = new Scanner(System.in);
7
8         System.out.print("Enter mass value: ");
9         double massValue = input.nextDouble();
10
11
12         System.out.print("Enter mass unit (MG/G/KG/LBS/ST): ");
13         String unit = input.next().toUpperCase();
14
15
16         System.out.print("Select new mass unit (MG/G/KG/LBS/ST): ");
17         String newUnit = input.next().toUpperCase();
18         input.close();
19
20         Mass mass;
21         if (unit.equalsIgnoreCase( anotherString: "MG")){
22             mass = new Milligram(massValue);
23             switch (newUnit){
24                 case "MG":
25                     System.out.println("\n" + massValue + " MG in MG: " + String.format("%.4f",mass.toMilligram()));
26                     break;
27                 case "G":
28                     System.out.println("\n" + massValue + " MG in G: " + String.format("%.4f",mass.toGram()));
29                     break;
30                 case "KG":
31                     System.out.println("\n" + massValue + " MG in KG: " + String.format("%.4f",mass.toKilogram()));
32                     break;
33                 case "LBS":
34                     System.out.println("\n" + massValue + " MG in LBS: " + String.format("%.4f",mass.toPound()));
35                     break;
36                 case "ST":
37                     System.out.println("\n" + massValue + " MG in ST: " + String.format("%.4f",mass.toStone()));
38                     break;
39             }
40         }else if (unit.equalsIgnoreCase( anotherString: "G")){
41             mass = new Gram(massValue);
42             switch (newUnit){
43                 case "MG":
44                     System.out.println("\n" + massValue + " G in MG: " + String.format("%.4f",mass.toMilligram()));
45                     break;
46                 case "G":
47                     System.out.println("\n" + massValue + " G in G: " + String.format("%.4f",mass.toGram()));
48                     break;
49                 case "KG":
50                     System.out.println("\n" + massValue + " G in KG: " + String.format("%.4f",mass.toKilogram()));
51                     break;
52                 case "LBS":
53                     System.out.println("\n" + massValue + " G in LBS: " + String.format("%.4f",mass.toPound()));
54                     break;
55                 case "ST":
56                     System.out.println("\n" + massValue + " G in ST: " + String.format("%.4f",mass.toStone()));
57                     break;
58             }
59         }
60     }
61 }
```

```
56.     }else if (unit.equalsIgnoreCase( anotherString: "KG")){
57.         mass = new Kilogram(massValue);
58.         switch (newUnit){
59.             case "MG":
60.                 System.out.println("\n" + massValue + " KG in MG: " + String.format("%.4f",mass.toMilligram()));
61.                 break;
62.             case "G":
63.                 System.out.println("\n" + massValue + " KG in G: " + String.format("%.4f",mass.toGram()));
64.                 break;
65.             case "KG":
66.                 System.out.println("\n" + massValue + " KG in KG: " + String.format("%.4f",mass.toKilogram()));
67.                 break;
68.             case "LBS":
69.                 System.out.println("\n" + massValue + " KG in LBS: " + String.format("%.4f",mass.toPound()));
70.                 break;
71.             case "ST":
72.                 System.out.println("\n" + massValue + " KG in ST: " + String.format("%.4f",mass.toStone()));
73.                 break;
74.         }
75.     }else if (unit.equalsIgnoreCase( anotherString: "LBS")){
76.         mass = new Pound(massValue);
77.         switch (newUnit){
78.             case "MG":
79.                 System.out.println("\n" + massValue + " LBS in MG: " + String.format("%.4f",mass.toMilligram()));
80.                 break;
81.             case "G":
82.                 System.out.println("\n" + massValue + " LBS in G: " + String.format("%.4f",mass.toGram()));
83.                 break;
84.             case "KG":
85.                 System.out.println("\n" + massValue + " LBS in KG: " + String.format("%.4f",mass.toKilogram()));
86.                 break;
87.             case "LBS":
88.                 System.out.println("\n" + massValue + " LBS in LBS: " + String.format("%.4f",mass.toPound()));
89.                 break;
90.             case "ST":
91.                 System.out.println("\n" + massValue + " LBS in ST: " + String.format("%.4f",mass.toStone()));
92.                 break;
93.         }
94.     }else if (unit.equalsIgnoreCase( anotherString: "ST")){
95.         mass = new Stone(massValue);
96.         switch (newUnit){
97.             case "MG":
98.                 System.out.println("\n" + massValue + " ST in MG: " + String.format("%.4f",mass.toMilligram()));
99.                 break;
100.            case "G":
101.                System.out.println("\n" + massValue + " ST in G: " + String.format("%.4f",mass.toGram()));
102.                break;
103.            case "KG":
104.                System.out.println("\n" + massValue + " ST in KG: " + String.format("%.4f",mass.toKilogram()));
105.                break;
106.            case "LBS":
107.                System.out.println("\n" + massValue + " ST in LBS: " + String.format("%.4f",mass.toPound()));
108.                break;
109.            case "ST":
110.                System.out.println("\n" + massValue + " ST in ST: " + String.format("%.4f",mass.toStone()));
111.                break;
112.        }
113.    }
114.}
115.}
```

Mass Class

```
1   ⚡ 6 usages 5 inheritors
2   ⚡ abstract class Mass {
3       26 usages
4       protected double measurement;
5
6   ⚡ 5 usages
7   ⚡ public Mass(double measurement) {this.measurement = measurement;}
8
9   ⚡ 5 usages 5 implementations
10  ⚡ public abstract double toMilligram();
11  ⚡ 5 usages 5 implementations
12  ⚡ public abstract double toGram();
13  ⚡ 5 usages 5 implementations
14  ⚡ public abstract double toKilogram();
15  ⚡ 5 usages 5 implementations
16  ⚡ public abstract double toPound();
17  ⚡ 5 usages 5 implementations
18  ⚡ public abstract double toStone();
19
20 } ━━━━━━━━
21
22 ⚡ 1 usage
23 ⚡ class Milligram extends Mass {
24     1 usage
25     public Milligram(double measurement) {super(measurement);}
26     5 usages
27     ⚡ public double toMilligram() {return measurement;}
28     5 usages
29     ⚡ public double toGram() {return measurement / 1000;}
30     5 usages
31     ⚡ public double toKilogram() {return measurement / 1000000;}
32     5 usages
33     ⚡ public double toPound() {return measurement / 453600;}
34     5 usages
35     ⚡ public double toStone() {return measurement / 6350000;}
36
37 } ━━━━━━━━
38
39 ⚡ 1 usage
40 ⚡ class Gram extends Mass {
41     1 usage
42     public Gram(double measurement) {super(measurement);}
43     5 usages
44     ⚡ public double toMilligram() {return measurement * 1000;}
45     5 usages
46     ⚡ public double toGram() {return measurement;}
47     5 usages
48     ⚡ public double toKilogram() {return measurement / 1000;}
49     5 usages
50     ⚡ public double toPound() {return measurement / 453.6;}
51     5 usages
52     ⚡ public double toStone() {return measurement / 6350;}
```

```
31     class Kilogram extends Mass {
32         1 usage
33         public Kilogram(double measurement) {super(measurement);}
34         5 usages
35         public double toMilligram() {return measurement * 1000000;}
36         5 usages
37         public double toGram() {return measurement * 1000;}
38         5 usages
39         public double toKilogram() {return measurement;}
40         5 usages
41         public double toPound() {return measurement * 2.205;}
42         5 usages
43         public double toStone() {return measurement / 6.35;}
44
45     }
46
47     class Pound extends Mass {
48         1 usage
49         public Pound(double measurement) {super(measurement);}
50         5 usages
51         public double toMilligram() {return measurement * 453600;}
52         5 usages
53         public double toGram() {return measurement * 453.6;}
54         5 usages
55         public double toKilogram() {return measurement / 2.265;}
56         5 usages
57         public double toPound() {return measurement;}
58         5 usages
59         public double toStone() {return measurement / 14;}
60
61     }
62
63     class Stone extends Mass {
64         1 usage
65         public Stone(double measurement) {super(measurement);}
66         5 usages
67         public double toMilligram() {return measurement * 6350000;}
68         5 usages
69         public double toGram() {return measurement * 6350;}
70         5 usages
71         public double toKilogram() {return measurement * 6.35;}
72         5 usages
73         public double toPound() {return measurement * 14;}
74         5 usages
75         public double toStone() {return measurement;}
76     }
```

Time

Main Class

```
 1 Main.java  Time.java *
 2
 3 import java.text.DecimalFormat;
 4 import java.text.NumberFormat;
 5 import java.util.Scanner;
 6
 7 no usages
 8 public class Main {
 9     no usages
10     public static void main(String[] args) throws Exception{
11         Scanner input = new Scanner(System.in);
12         NumberFormat formatter = new DecimalFormat( pattern: "#0.0");
13
14         System.out.print("Enter Time value: ");
15         double timeValue = input.nextDouble();
16
17         System.out.println("1. (S) Second \n2. (Min) Minute \n3. (H) Hour \n4. (D) Day\n5. (W) Week\n6. (M) Month \n7. (Y) Year \n8. (Dec) Decade");
18
19         System.out.print("Select unit(S/Min/H/D/W/M/Y/Dec) (for old Time): ");
20         String unit = input.nextLine().toUpperCase();
21
22         System.out.print("Select unit(S/Min/H/D/W/M/Y/Dec) (for new Time): ");
23         String unit2 = input.nextLine().toUpperCase();
24         input.close();
25
26         TIME time;
27         if (unit.equalsIgnoreCase( anotherString: "S")){
28             time = new Second(timeValue);
29
30             switch (unit2){
31                 case "MIN":
32                     System.out.println("\n" + timeValue + " Second in Minute: " + String.format("%.2f", time.toMin()));
33                     break;
34                 case "H":
35                     System.out.println("\n" + timeValue + " Second in Hour: " + String.format("%.2f", time.toHour()));
36                     break;
37                 case "D":
38                     System.out.println("\n" + timeValue + " Second in Day: " + String.format("%.2f", time.toDay()));
39                     break;
40                 case "W":
41                     System.out.println("\n" + timeValue + " Second in Week: " + String.format("%.2f", time.toWeek()));
42                     break;
43                 case "M":
44                     System.out.println("\n" + timeValue + " Second in Month: " + String.format("%.2f", time.toMonth()));
45                     break;
46                 case "Y":
47                     System.out.println("\n" + timeValue + " Second in Year: " + String.format("%.2f", time.toYear()));
48                     break;
49                 case "DEC":
50                     System.out.println("\n" + timeValue + " Second in Decade: " + String.format("%.2f", time.toDecade()));
51                     break;
52             }
53         }else if (unit.equalsIgnoreCase( anotherString: "Min")){
54             time = new Second(timeValue);
55             switch (unit2){
56                 case "S":
57                     System.out.println("\n" + timeValue + " Min in Second: " + String.format("%.2f", time.toSecond()));
58                     break;
59                 case "H":
60                     System.out.println("\n" + timeValue + " Min in Hour: " + String.format("%.2f", time.toHour()));
61                     break;
62                 case "D":
63                     System.out.println("\n" + timeValue + " Min in Day: " + String.format("%.2f", time.toDay()));
64                     break;
65                 case "W":
66                     System.out.println("\n" + timeValue + " Min in Week: " + String.format("%.2f", time.toWeek()));


```

```

62             break;
63         case "M":
64             System.out.println("\n" + timeValue + " Min in Month: " + String.format("%.2f", time.toMonth()));
65             break;
66         case "Y":
67             System.out.println("\n" + timeValue + " Min in Year: " + String.format("%.2f", time.toYear()));
68             break;
69         case "DEC":
70             System.out.println("\n" + timeValue + " Min in Decade: " + String.format("%.2f", time.toDecade()));
71             break;
72     }
73 }
74 } else if (unit.equalsIgnoreCase( anotherString: "H")){
75     time = new Second(timeValue);
76     switch (unit2){
77         case "S":
78             System.out.println("\n" + timeValue + " Hour in Second: " + String.format("%.2f", time.toSecond()));
79             break;
80         case "MIN":
81             System.out.println("\n" + timeValue + " Hour in Minute: " + String.format("%.2f", time.toMin()));
82             break;
83         case "D":
84             System.out.println("\n" + timeValue + " Hour in Day: " + String.format("%.2f", time.toDay()));
85             break;
86         case "W":
87             System.out.println("\n" + timeValue + " Hour in Week: " + String.format("%.2f", time.toWeek()));
88             break;
89         case "M":
90             System.out.println("\n" + timeValue + " Hour in Month: " + String.format("%.2f", time.toMonth()));
91             break;
92         case "Y":
93             System.out.println("\n" + timeValue + " Hour in Year: " + String.format("%.2f", time.toYear()));
94             break;
95         case "DEC":
96             System.out.println("\n" + timeValue + " Hour in Decade: " + String.format("%.2f", time.toDecade()));
97             break;
98     }
99 } else if (unit.equalsIgnoreCase( anotherString: "D")){
100    time = new Second(timeValue);
101    switch (unit2){
102        case "S":
103            System.out.println("\n" + timeValue + " Day in Second: " + String.format("%.2f", time.toSecond()));
104            break;
105        case "MIN":
106            System.out.println("\n" + timeValue + " Day in Minute: " + String.format("%.2f", time.toMin()));
107            break;
108        case "H":
109            System.out.println("\n" + timeValue + " Day in Hour: " + String.format("%.2f", time.toHour()));
110            break;
111        case "W":
112            System.out.println("\n" + timeValue + " Day in Week: " + String.format("%.2f", time.toWeek()));
113            break;
114        case "M":
115            System.out.println("\n" + timeValue + " Day in Month: " + String.format("%.2f", time.toMonth()));
116            break;
117        case "Y":
118            System.out.println("\n" + timeValue + " Day in Year: " + String.format("%.2f", time.toYear()));
119            break;
120        case "DEC":
121            System.out.println("\n" + timeValue + " Day in Decade: " + String.format("%.2f", time.toDecade()));
122            break;
123     }
124 } else if (unit.equalsIgnoreCase( anotherString: "W")){
125     time = new Second(timeValue);

```

```

125     switch (unit2){
126         case "S":
127             System.out.println("\n" + timeValue + " Week in Second: " + String.format("%.2f", time.toSecond()));
128             break;
129         case "MIN":
130             System.out.println("\n" + timeValue + " Week in Minute: " + String.format("%.2f", time.toMin()));
131             break;
132         case "H":
133             System.out.println("\n" + timeValue + " Week in Hour: " + String.format("%.2f", time.toHour()));
134             break;
135         case "D":
136             System.out.println("\n" + timeValue + " Week in Day: " + String.format("%.2f", time.toDay()));
137             break;
138         case "M":
139             System.out.println("\n" + timeValue + " Week in Month: " + String.format("%.2f", time.toMonth()));
140             break;
141         case "Y":
142             System.out.println("\n" + timeValue + " Week in Year: " + String.format("%.2f", time.toYear()));
143             break;
144         case "DEC":
145             System.out.println("\n" + timeValue + " Week in Decade: " + String.format("%.2f", time.toDecade()));
146             break;
147     }
148 }else if (unit.equalsIgnoreCase( anotherString: "M")) {
149     time = new Second(timeValue);
150     switch (unit2) {
151         case "S":
152             System.out.println("\n" + timeValue + " Month in Second: " + String.format("%.2f", time.toSecond()));
153             break;
154         case "MIN":
155             System.out.println("\n" + timeValue + " Month in Minute: " + String.format("%.2f", time.toMin()));
156             break;
157         case "H":
158             System.out.println("\n" + timeValue + " Month in Hour: " + String.format("%.2f", time.toHour()));
159             break;
160         case "D":
161             System.out.println("\n" + timeValue + " Month in Day: " + String.format("%.2f", time.toDay()));
162             break;
163         case "W":
164             System.out.println("\n" + timeValue + " Month in Week: " + String.format("%.2f", time.toWeek()));
165             break;
166         case "Y":
167             System.out.println("\n" + timeValue + " Month in Year: " + String.format("%.2f", time.toYear()));
168             break;
169         case "DEC":
170             System.out.println("\n" + timeValue + " Month in Decade: " + String.format("%.2f", time.toDecade()));
171             break;
172     }
173 }else if (unit.equalsIgnoreCase( anotherString: "Y")) {
174     time = new Second(timeValue);
175     switch (unit2) {
176         case "S":
177             System.out.println("\n" + timeValue + " Year in Second: " + String.format("%.2f", time.toSecond()));
178             break;
179         case "MIN":
180             System.out.println("\n" + timeValue + " Year in Minute: " + String.format("%.2f", time.toMin()));
181             break;
182         case "H":
183             System.out.println("\n" + timeValue + " Year in Hour: " + String.format("%.2f", time.toHour()));
184             break;
185         case "D":
186             System.out.println("\n" + timeValue + " Year in Day: " + String.format("%.2f", time.toDay()));
187             break;
188     }
189 }
```

```
190         case "W":
191             System.out.println("\n" + timeValue + " Year in Week: " + String.format("%.2f", time.toWeek()));
192             break;
193         case "M":
194             System.out.println("\n" + timeValue + " Year in Month: " + String.format("%.2f", time.toMonth()));
195             break;
196         case "DEC":
197             System.out.println("\n" + timeValue + " Year in Decade: " + String.format("%.2f", time.toDecade()));
198             break;
199     }
200     }else if (unit.equalsIgnoreCase( anotherString: "Dec")){
201         time = new Second(timeValue);
202         switch (unitz) {
203             case "S":
204                 System.out.println("\n" + timeValue + " Decade in Second: " + String.format("%.2f", time.toSecond()));
205                 break;
206             case "MIN":
207                 System.out.println("\n" + timeValue + " Decade in Minute: " + String.format("%.2f", time.toMin()));
208                 break;
209             case "H":
210                 System.out.println("\n" + timeValue + " Decade in Hour: " + String.format("%.2f", time.toHour()));
211                 break;
212             case "D":
213                 System.out.println("\n" + timeValue + " Decade in Day: " + String.format("%.2f", time.toDay()));
214                 break;
215             case "W":
216                 System.out.println("\n" + timeValue + " Decade in Week: " + String.format("%.2f", time.toWeek()));
217                 break;
218             case "M":
219                 System.out.println("\n" + timeValue + " Decade in Month: " + String.format("%.2f", time.toMonth()));
220                 break;
221             case "Y":
222                 System.out.println("\n" + timeValue + " Decade in Year: " + String.format("%.2f", time.toYear()));
223                 break;
224         }
225     }else{
226         System.out.print("Invalid time| unit");
227         return;
228     }
229 }
230 }
```

Time Class

```
Main.java × Time.java ×
abstract class Time {
    protected double value;
    public Time(double value) {
        this.value = value;
    }
    public abstract double toSecond();
    public abstract double toMin();
    public abstract double toHour();
    public abstract double toDay();
    public abstract double toWeek();
    public abstract double toMonth();
    public abstract double toYear();
}
```

```
22 ①    public abstract double toDecade();  
23  
24  
25  }  
     8 usages  
26  class Second extends Time {  
     8 usages  
27  public Second(double value) { super(value); }  
28  
     7 usages  
31 ②    public double toSecond() { return value; }  
32  
     7 usages  
35 ③    public double toMin() { return (value/60); }  
36  
     7 usages  
39 ④    public double toHour() { return (value/3600); }  
40  
     7 usages  
42 ⑤    public double toDay() { return (value/86400); }  
43  
     7 usages  
45 ⑥    public double toWeek() { return value/604800; }  
46  
     7 usages  
48 ⑦    public double toMonth() { return value/2628000; }  
49  
     7 usages  
51 ⑧    public double toYear() { return value/31536000; }  
52  
     7 usages  
54 ⑨    public double toDecade() { return value/315360000; }  
55  
56  }  
     no usages  
59  class Minute extends Time {
```

```
60     public Minute(double value) { super(value); }
63
64 ①↑    7 usages
64 ①↑    public double toSecond() { return value*60; }
67
68 ①↑    7 usages
68 ①↑    public double toMin() { return value; }
71
72 ①↑    7 usages
72 ①↑    public double toHour() { return (value/60); }
75 ①↑    7 usages
75 ①↑    public double toDay() { return value/1440; }
78 ①↑    7 usages
78 ①↑    public double toWeek() { return value/10080; }
81 ①↑    7 usages
81 ①↑    public double toMonth() { return value/43829.1; }
84 ①↑    7 usages
84 ①↑    public double toYear() { return value/525949; }
87 ①↑    7 usages
87 ①↑    public double toDecade() { return value/5259492; }
90
91
92    }
92    no usages
93    class Hour extends Time {
93    no usages
94    public Hour(double value) { super(value); }
97
98 ①↑    7 usages
98 ①↑    public double toSecond() { return value*3600; }
101
```

```
102 ↑    public double toMin() { return value*60; }
105
106 ↑    public double toHour() { return value; }
109 ↑    public double toDay() { return value/24; }
112 ↑    public double toWeek() { return value/168; }
115 ↑    public double toMonth() { return value/730.484; }
118 ↑    public double toYear() { return value/8760; }
121 ↑    public double toDecade() { return value/87600; }
124
125
126 }
127 class Day extends Time {
128     public Day(double value) { super(value); }
131
132 ↑    public double toSecond() { return value*86400; }
135
136 ↑    public double toMin() { return (value*1440); }
139
140 ↑    public double toHour() { return (value*24); }
```

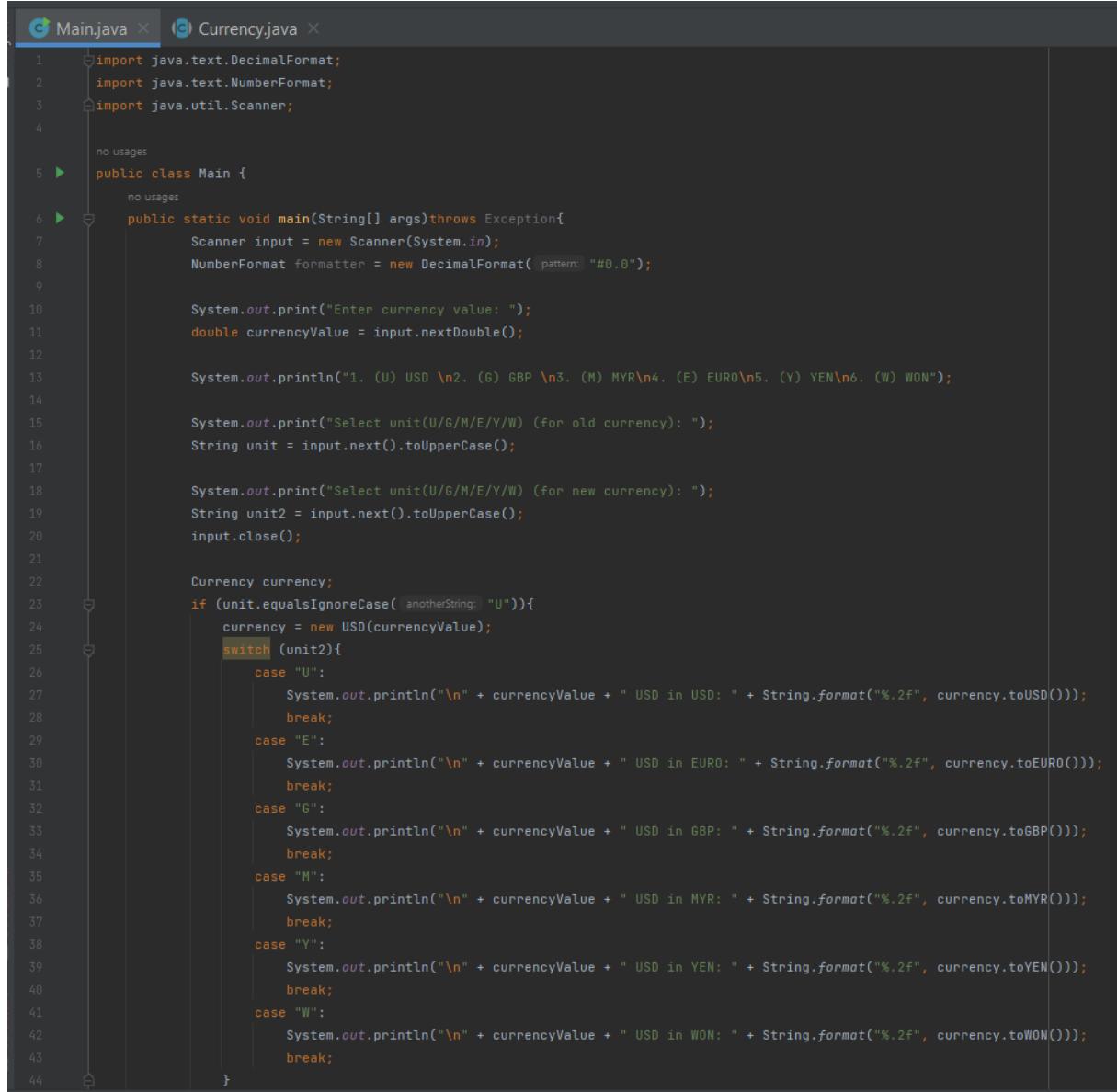
```
143 ↑ ☐ public double toDay() { return value; }
    7 usages
146 ↑ ☐ public double toWeek() { return value/7; }
    7 usages
149 ↑ ☐ public double toMonth() { return value/30; }
    7 usages
152 ↑ ☐ public double toYear() { return value/365; }
    7 usages
155 ↑ ☐ public double toDecade() { return value/3650; }
158
159 ☐ no usages
160 class Week extends Time {
    no usages
161 ☐ public Week(double value) { super(value); }
164
165 ↑ ☐ public double toSecond() { return value*604800; }
168
169 ↑ ☐ public double toMin() { return value*10080; }
172
173 ↑ ☐ public double toHour() { return value*168; }
    7 usages
176 ↑ ☐ public double toDay() { return value*7; }
    7 usages
179 ↑ ☐ public double toWeek() { return value; }
    7 usages
182 ↑ ☐ public double toMonth() { return value/4; }
    7 usages
185 ↑ ☐ public double toYear() { return value/52; }
```

```
188 ⚡}     public double toDecade() { return value/520; }
191
192 ⚡}
193     no usages
194     class Month extends Time {
195         no usages
196         public Month(double value) { super(value); }
197
198 ⚡}     7 usages
199     public double toSecond() { return value*2592000; }
200
201     7 usages
202 ⚡}     7 usages
203     public double toMin() { return value*43200; }
204
205     7 usages
206 ⚡}     7 usages
207     public double toHour() { return value*720; }
208
209 ⚡}     7 usages
210     public double toDay() { return value*30; }
211
212 ⚡}     7 usages
213     public double toWeek() { return value*4; }
214
215 ⚡}     7 usages
216     public double toMonth() { return value; }
217
218 ⚡}     7 usages
219     public double toYear() { return value/12; }
220
221 ⚡}     7 usages
222     public double toDecade() { return value/120; }
223
224 ⚡}
225     no usages
226     class Year extends Time {
```

```
227     public Year(double value) { super(value); }
230
231  ↗    7 usages
231  ↗    public double toSecond() { return value*31536000; }
234
235  ↗    7 usages
235  ↗    public double toMin() { return value*525600; }
238
239  ↗    7 usages
239  ↗    public double toHour() { return value*8760; }
242
242  ↗    7 usages
242  ↗    public double toDay() { return value*365; }
245
245  ↗    7 usages
245  ↗    public double toWeek() { return value*52; }
248
248  ↗    7 usages
248  ↗    public double toMonth() { return value*12; }
251
251  ↗    7 usages
251  ↗    public double toYear() { return value; }
254
254  ↗    7 usages
254  ↗    public double toDecade() { return value/10; }
257
258  ↘}
258  ↘ no usages
259  ↗ class Decade extends Time {
259  ↗ no usages
260  ↗    public Decade(double value) { super(value); }
263
264  ↗    7 usages
264  ↗    public double toSecond() { return value*31536000; }
267
268  ↗    7 usages
268  ↗    public double toMin() { return (value*5259492); }
271
271  ↗    7 usages
272  ↗    public double toHour() { return (value*87600); }
275
275  ↗    7 usages
275  ↗    public double toDay() { return value*3650; }
278
278  ↗    7 usages
278  ↗    public double toWeek() { return value* 521; }
281
281  ↗    7 usages
281  ↗    public double toMonth() { return value*120; }
284
284  ↗    7 usages
284  ↗    public double toYear() { return value*10; }
287
287  ↗    7 usages
287  ↗    public double toDecade() { return value; }
290
291  ↘}
```

Currency

Main Class



```
1 import java.text.DecimalFormat;
2 import java.text.NumberFormat;
3 import java.util.Scanner;
4
5 public class Main {
6     public static void main(String[] args) throws Exception{
7         Scanner input = new Scanner(System.in);
8         NumberFormat formatter = new DecimalFormat( pattern: "#0.0");
9
10        System.out.print("Enter currency value: ");
11        double currencyValue = input.nextDouble();
12
13        System.out.println("1. (U) USD \n2. (G) GBP \n3. (M) MYR\n4. (E) EURO\n5. (Y) YEN\n6. (W) WON");
14
15        System.out.print("Select unit(U/G/M/E/Y/W) (for old currency): ");
16        String unit = input.next().toUpperCase();
17
18        System.out.print("Select unit(U/G/M/E/Y/W) (for new currency): ");
19        String unit2 = input.next().toUpperCase();
20        input.close();
21
22        Currency currency;
23        if (unit.equalsIgnoreCase( anotherString: "U")){
24            currency = new USD(currencyValue);
25            switch (unit2){
26                case "U":
27                    System.out.println("\n" + currencyValue + " USD in USD: " + String.format("%.2f", currency.toUSD()));
28                    break;
29                case "E":
30                    System.out.println("\n" + currencyValue + " USD in EURO: " + String.format("%.2f", currency.toEURO()));
31                    break;
32                case "G":
33                    System.out.println("\n" + currencyValue + " USD in GBP: " + String.format("%.2f", currency.toGBP()));
34                    break;
35                case "M":
36                    System.out.println("\n" + currencyValue + " USD in MYR: " + String.format("%.2f", currency.toMYR()));
37                    break;
38                case "Y":
39                    System.out.println("\n" + currencyValue + " USD in YEN: " + String.format("%.2f", currency.toYEN()));
39                    break;
38                case "W":
39                    System.out.println("\n" + currencyValue + " USD in WON: " + String.format("%.2f", currency.toWON()));
39                    break;
40            }
41        }
42    }
43}
44}
```

```
        }else if (unit.equalsIgnoreCase( anotherString: "E")){
            currency = new EURO(currencyValue);
            switch (unit2){
                case "U":
                    System.out.println("\n" + currencyValue + " EURO in USD: " + String.format("%.2f", currency.toUSD()));
                    break;
                case "E":
                    System.out.println("\n" + currencyValue + " EURO in EURO: " + String.format("%.2f", currency.toEURO()));
                    break;
                case "G":
                    System.out.println("\n" + currencyValue + " EURO in GBP: " + String.format("%.2f", currency.toGBP()));
                    break;
                case "M":
                    System.out.println("\n" + currencyValue + " EURO in MYR: " + String.format("%.2f", currency.toMYR()));
                    break;
                case "Y":
                    System.out.println("\n" + currencyValue + " EURO in YEN: " + String.format("%.2f", currency.toYEN()));
                    break;
                case "W":
                    System.out.println("\n" + currencyValue + " EURO in WON: " + String.format("%.2f", currency.toWON()));
                    break;
            }
        }else if (unit.equalsIgnoreCase( anotherString: "G")){
            currency = new GBP(currencyValue);
            switch (unit2){
                case "U":
                    System.out.println("\n" + currencyValue + " GBP in USD: " + String.format("%.2f", currency.toUSD()));
                    break;
                case "E":
                    System.out.println("\n" + currencyValue + " GBP in EURO: " + String.format("%.2f", currency.toEURO()));
                    break;
                case "G":
                    System.out.println("\n" + currencyValue + " GBP in GBP: " + String.format("%.2f", currency.toGBP()));
                    break;
                case "M":
                    System.out.println("\n" + currencyValue + " GBP in MYR: " + String.format("%.2f", currency.toMYR()));
                    break;
                case "Y":
                    System.out.println("\n" + currencyValue + " GBP in YEN: " + String.format("%.2f", currency.toYEN()));
                    break;
                case "W":
                    System.out.println("\n" + currencyValue + " GBP in WON: " + String.format("%.2f", currency.toWON()));
                    break;
            }
        }
```

```
89     }else if (unit.equalsIgnoreCase( anotherString: "M")){
90         currency = new MYR(currencyValue);
91         switch (unit2){
92             case "U":
93                 System.out.println("\n" + currencyValue + " MYR in USD: " + String.format("%.2f", currency.toUSD()));
94                 break;
95             case "E":
96                 System.out.println("\n" + currencyValue + " MYR in EURO: " + String.format("%.2f", currency.toEURO()));
97                 break;
98             case "G":
99                 System.out.println("\n" + currencyValue + " MYR in GBP: " + String.format("%.2f", currency.toGBP()));
100                break;
101            case "M":
102                System.out.println("\n" + currencyValue + " MYR in MYR: " + String.format("%.2f", currency.toMYR()));
103                break;
104            case "Y":
105                System.out.println("\n" + currencyValue + " MYR in YEN: " + String.format("%.2f", currency.toYEN()));
106                break;
107            case "W":
108                System.out.println("\n" + currencyValue + " MYR in WON: " + String.format("%.2f", currency.toWON()));
109                break;
110        }
111    }else if (unit.equalsIgnoreCase( anotherString: "Y")){
112        currency = new YEN(currencyValue);
113        switch (unit2){
114            case "U":
115                System.out.println("\n" + currencyValue + " YEN in USD: " + String.format("%.2f", currency.toUSD()));
116                break;
117            case "E":
118                System.out.println("\n" + currencyValue + " YEN in EURO: " + String.format("%.2f", currency.toEURO()));
119                break;
120            case "G":
121                System.out.println("\n" + currencyValue + " YEN in GBP: " + String.format("%.2f", currency.toGBP()));
122                break;
123            case "M":
124                System.out.println("\n" + currencyValue + " YEN in MYR: " + String.format("%.2f", currency.toMYR()));
125                break;
126            case "Y":
127                System.out.println("\n" + currencyValue + " YEN in YEN: " + String.format("%.2f", currency.toYEN()));
128                break;
129            case "W":
130                System.out.println("\n" + currencyValue + " YEN in WON: " + String.format("%.2f", currency.toWON()));
131                break;
132        }
133    }
```

```
        }
    }else if (unit.equalsIgnoreCase( anotherString: "W")){
        currency = new WON(currencyValue);
        switch (unit2){
            case "U":
                System.out.println("\n" + currencyValue + " WON in USD: " + String.format("%.2f", currency.toUSD()));
                break;
            case "E":
                System.out.println("\n" + currencyValue + " WON in EURO: " + String.format("%.2f", currency.toEURO()));
                break;
            case "G":
                System.out.println("\n" + currencyValue + " WON in GBP: " + String.format("%.2f", currency.toGBP()));
                break;
            case "M":
                System.out.println("\n" + currencyValue + " WON in MYR: " + String.format("%.2f", currency.toMYR()));
                break;
            case "Y":
                System.out.println("\n" + currencyValue + " WON in YEN: " + String.format("%.2f", currency.toYEN()));
                break;
            case "W":
                System.out.println("\n" + currencyValue + " WON in WON: " + String.format("%.2f", currency.toWON()));
                break;
        }
    }else{
        System.out.print("Invalid currency unit");
        return;
    }
}
}
```

Currency Class

```
abstract class Currency {
    37 usages
    protected double value;

    6 usages
    public Currency(double value) { this.value = value; }

    6 usages 6 implementations
    public abstract double toUSD();

    6 usages 6 implementations
    public abstract double toEURO();

    6 usages 6 implementations
    public abstract double toMYR();

    6 usages 6 implementations
    public abstract double toGBP();

    6 usages 6 implementations
    public abstract double toYEN();

    6 usages 6 implementations
    public abstract double toWON();

}

1 usage
class USD extends Currency{
    1 usage
    public USD(double value) { super(value); }

    6 usages
    public double toUSD() { return value; }

    6 usages
    public double toEURO() { return value * 0.92249877; }

    6 usages
    public double toGBP() { return value * 0.81113317; }

    6 usages
    public double toMYR(){return value * 4.4016427; }

    6 usages
    public double toYEN() { return value * 131.63494; }

    6 usages
    public double toWON() { return value * 1299.6995; }
}
```

```
class EURO extends Currency{
    1 usage
    public EURO(double value) { super(value); }
    6 usages
    public double toUSD() { return value * 1.0839245; }
    6 usages
    public double toEURO() { return value; }

    6 usages
    public double toGBP() { return value * 0.87922779; }

    6 usages
    public double toMYR() { return value * 4.7711903; }
    6 usages
    public double toYEN() { return value * 142.68488; }
    6 usages
    public double toWON() { return value * 1408.5749; }
}

1 usage
class GBP extends Currency {
    1 usage
    public GBP(double value) { super(value); }

    6 usages
    public double toUSD() { return value * 1.2327927; }

    6 usages
    public double toEURO() { return value * 1.1373937; }

    6 usages
    public double toGBP() { return value; }

    6 usages
    public double toMYR() { return value * 5.4268354; }

    6 usages
    public double toYEN() { return value * 162.28008; }

    6 usages
    public double toWON() { return value * 1601.9348; }
}
```

```
class MYR extends Currency {
    1 usage
    public MYR(double value) { super(value); }
    6 usages
    public double toUSD() { return value * 0.22719659; }
    6 usages
    public double toEURO() { return value * 0.20961126; }

    6 usages
    public double toGBP() { return value * 0.18429672; }

    6 usages
    public double toMYR() { return value; }

    6 usages
    public double toYEN() { return value * 29.909007; }

    6 usages
    public double toWON() { return value * 295.2838; }

}

1 usage
class YEN extends Currency {
    1 usage
    public YEN(double value) { super(value); }

    6 usages
    public double toUSD() { return value * 0.0075911457; }

    6 usages
    public double toEURO() { return value * 0.0070035179; }

    6 usages
    public double toGBP() { return value * 0.006157825; }

    6 usages
    public double toMYR() { return value * 0.033410559; }

    6 usages
    public double toYEN() { return value; }

    6 usages
    public double toWON() { return value * 9.8641199; }

}
```

```
class WON extends Currency {  
    1 usage  
    public WON(double value) { super(value); }  
  
    6 usages  
    public double toUSD() { return value * 0.00076963052; }  
  
    6 usages  
    public double toEURO() { return value * 0.00071010101; }  
  
    6 usages  
    public double toGBP() { return value * 0.00062434176; }  
  
    6 usages  
    public double toMYR() { return value * 0.0033875979; }  
  
    6 usages  
    public double toYEN() { return value * 0.10137495; }  
  
    6 usages  
    public double toWON() { return value; }  
}
```

Pressure

Main class

```
>Main.java × Pressure.java ×
1 import java.text.DecimalFormat;
2 import java.text.NumberFormat;
3 import java.util.*;
4
5 public class Main {
6     public static void main(String[] args) throws Exception {
7         Scanner input = new Scanner(System.in);
8         NumberFormat formatter = new DecimalFormat("##0.0000");
9
10        System.out.print("Enter a Pressure Value: ");
11        double pressureValue = input.nextDouble();
12
13        System.out.print("Enter the Pressure Unit (PA/ATM): ");
14        String unit1 = input.next().toUpperCase();
15
16        System.out.print("Enter the Pressure Unit conversion (PA/ATM): ");
17        String unit2 = input.next().toUpperCase();
18        input.close();
19
20        Pressure pressure;
21        if (unit1.equalsIgnoreCase("PA")) {
22            pressure = new Pascal(pressureValue);
23            switch (unit2) {
24                case "PA":
25                    System.out.println("Pressure in Pascal to Pascal: " + formatter.format(pressure.toPascal()));
26                    break;
27                case "ATM":
28                    System.out.println("Pressure in Pascal to Atmosphere: " + formatter.format(pressure.toAtmosphere()));
29            }
30        } else if (unit1.equalsIgnoreCase("ATM")) {
31            pressure = new Atmosphere(pressureValue);
32            switch (unit2) {
33                case "PA":
34                    System.out.println("Pressure in Atmosphere to Pascal: " + formatter.format(pressure.toPascal()));
35                    break;
36                case "ATM":
37                    System.out.println("Pressure in Atmosphere to Atmosphere: " + formatter.format(pressure.toAtmosphere()));
38            }
39        } else {
40            System.out.print("Invalid Pressure Unit");
41            return;
42        }
43    }
44}
```

```
Pressure pressure;
if (unit1.equalsIgnoreCase("PA")) {
    pressure = new Pascal(pressureValue);
    switch (unit2) {
        case "PA":
            System.out.println("Pressure in Pascal to Pascal: " + formatter.format(pressure.toPascal()));
            break;
        case "ATM":
            System.out.println("Pressure in Pascal to Atmosphere: " + formatter.format(pressure.toAtmosphere()));
    }
} else if (unit1.equalsIgnoreCase("ATM")) {
    pressure = new Atmosphere(pressureValue);
    switch (unit2) {
        case "PA":
            System.out.println("Pressure in Atmosphere to Pascal: " + formatter.format(pressure.toPascal()));
            break;
        case "ATM":
            System.out.println("Pressure in Atmosphere to Atmosphere: " + formatter.format(pressure.toAtmosphere()));
    }
} else {
    System.out.print("Invalid Pressure Unit");
    return;
}
}
```

Pressure class

```
1  Main.java x  Pressure.java x
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```

```
1  abstract class Pressure {
2      protected double value;
3
4      public Pressure(double value) {
5          this.value = value;
6      }
7
8      public abstract double toPascal();
9
10     public abstract double toAtmosphere();
11 }
12
13 class Pascal extends Pressure {
14     public Pascal(double value) {
15         super(value);
16     }
17
18     public double toPascal() {
19         return value;
20     }
21
22     public double toAtmosphere() {
23         return value / 101325;
24     }
25 }
26
27 class Atmosphere extends Pressure {
28     public Atmosphere(double value) {
29         super(value);
30     }
31
32     public double toPascal() {
33         return value * 101325;
34     }
35
36     public double toAtmosphere() {
37         return value;
38     }
39 }
40
```

Area

Main class

```
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.Scanner;

no usages
public class Main {
    no usages

    public static void main(String[] args) throws Exception{
        Scanner input = new Scanner(System.in);
        NumberFormat formatter = new DecimalFormat('###,##0.0');

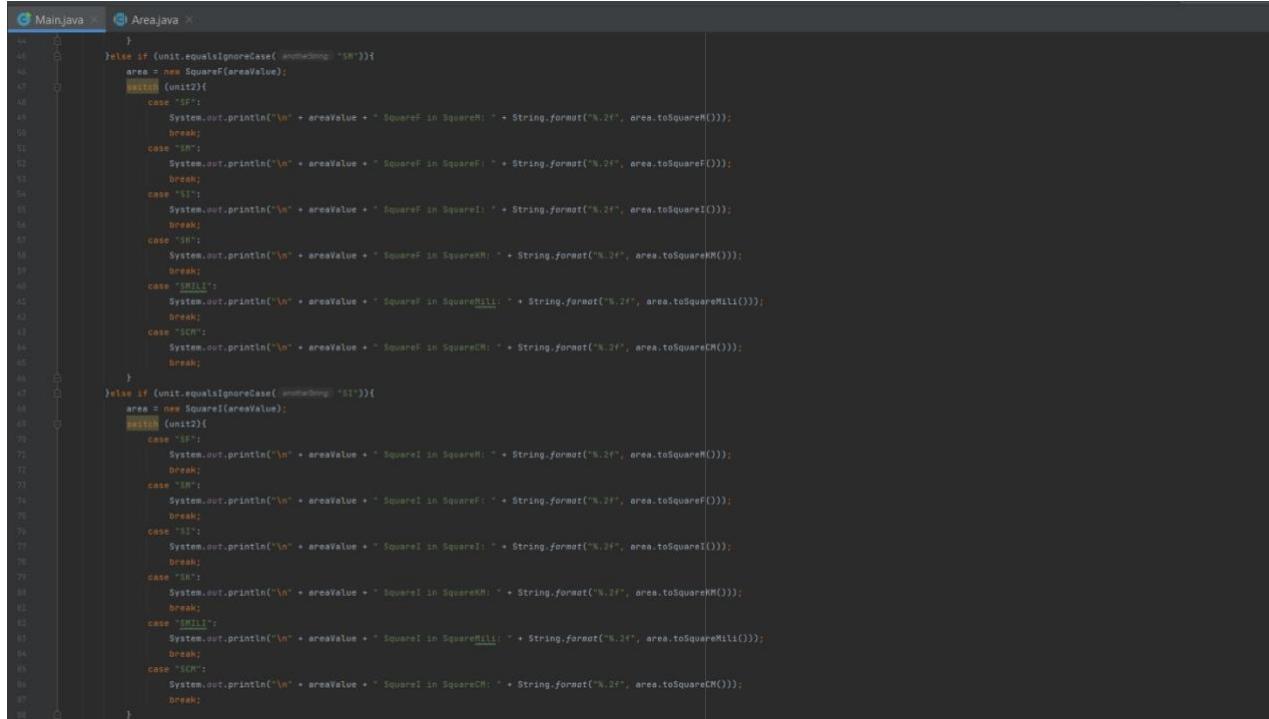
        System.out.print("Enter currency value: ");
        double areaValue = input.nextDouble();

        System.out.println("1. (SF) Square Meter \n2. (SI) Square Inches \n3. (SKM) Square Kilometer\n4. (SF) Square Foot\n5. (SMILLI) Square Millimeter\n6. (SCM) Square Centimeter");

        System.out.print("Select unit(SF/SI/SKM/SF/SMILLI/SCM) (for old currency): ");
        String unit = input.next().toUpperCase();

        System.out.print("Select unit(SF/SI/SKM/SF/SMILLI/SCM) (for new currency): ");
        String unit2 = input.next().toUpperCase();
        input.close();

        Area area;
        if (unit.equalsIgnoreCase("SF")){
            area = new SquareM(areaValue);
        }
        switch (unit2){
            case "SF":
                System.out.println("\n" + areaValue + " SquareM in SquareM: " + String.format("%.2f", area.toSquareM()));
                break;
            case "SM":
                System.out.println("\n" + areaValue + " SquareM in SquareF: " + String.format("%.2f", area.toSquareF()));
                break;
            case "SI":
                System.out.println("\n" + areaValue + " SquareM in SquareI: " + String.format("%.2f", area.toSquareI()));
                break;
            case "SKM":
                System.out.println("\n" + areaValue + " SquareM in SquareKM: " + String.format("%.2f", area.toSquareKM()));
                break;
            case "SMILLI":
                System.out.println("\n" + areaValue + " SquareM in SquareMilli: " + String.format("%.2f", area.toSquareMilli()));
                break;
            case "SCM":
                System.out.println("\n" + areaValue + " SquareM in SquareCM: " + String.format("%.2f", area.toSquareCM()));
                break;
        }
    }
}
```



```
44
45
46    else if (unit.equalsIgnoreCase("SM")){
47        area = new SquareF(areaValue);
48    }
49    switch (unit2){
50        case "SF":
51            System.out.println("\n" + areaValue + " SquareF in SquareM: " + String.format("%.2f", area.toSquareM()));
52            break;
53        case "SM":
54            System.out.println("\n" + areaValue + " SquareF in SquareF: " + String.format("%.2f", area.toSquareF()));
55            break;
56        case "SI":
57            System.out.println("\n" + areaValue + " SquareF in SquareI: " + String.format("%.2f", area.toSquareI()));
58            break;
59        case "SKM":
60            System.out.println("\n" + areaValue + " SquareF in SquareKM: " + String.format("%.2f", area.toSquareKM()));
61            break;
62        case "SMILLI":
63            System.out.println("\n" + areaValue + " SquareF in SquareMilli: " + String.format("%.2f", area.toSquareMilli()));
64            break;
65        case "SCM":
66            System.out.println("\n" + areaValue + " SquareF in SquareCM: " + String.format("%.2f", area.toSquareCM()));
67            break;
68    }
69
70 }else if (unit.equalsIgnoreCase("SI")){
71     area = new SquareI(areaValue);
72 }
73 switch (unit2){
74     case "SF":
75         System.out.println("\n" + areaValue + " SquareI in SquareM: " + String.format("%.2f", area.toSquareM()));
76         break;
77     case "SM":
78         System.out.println("\n" + areaValue + " SquareI in SquareF: " + String.format("%.2f", area.toSquareF()));
79         break;
80     case "SI":
81         System.out.println("\n" + areaValue + " SquareI in SquareI: " + String.format("%.2f", area.toSquareI()));
82         break;
83     case "SKM":
84         System.out.println("\n" + areaValue + " SquareI in SquareKM: " + String.format("%.2f", area.toSquareKM()));
85         break;
86     case "SMILLI":
87         System.out.println("\n" + areaValue + " SquareI in SquareMilli: " + String.format("%.2f", area.toSquareMilli()));
88         break;
89     case "SCM":
90         System.out.println("\n" + areaValue + " SquareI in SquareCM: " + String.format("%.2f", area.toSquareCM()));
91         break;
92 }
```

```

Main.java X Area.java X
89 }else if (unit.equalsIgnoreCase( anotherString: "SK")){
90     area = new SquareKM(areaValue);
91     switch (unit2){
92         case "SF":
93             System.out.println("\n" + areaValue + " SquareKM in SquareF: " + String.format("%.2f", area.toSquareF()));
94             break;
95         case "SM":
96             System.out.println("\n" + areaValue + " SquareKM in SquareM: " + String.format("%.2f", area.toSquareM()));
97             break;
98         case "SI":
99             System.out.println("\n" + areaValue + " SquareKM in SquareI: " + String.format("%.2f", area.toSquareI()));
100            break;
101        case "SK":
102            System.out.println("\n" + areaValue + " SquareKM in SquareKM: " + String.format("%.2f", area.toSquareKM()));
103            break;
104        case "SMLI":
105            System.out.println("\n" + areaValue + " SquareKM in SquareMili: " + String.format("%.2f", area.toSquareMili()));
106            break;
107        case "SCM":
108            System.out.println("\n" + areaValue + " SquareKM in SquareCM: " + String.format("%.2f", area.toSquareCM()));
109            break;
110    }
111 }else if (unit.equalsIgnoreCase( anotherString: "SMLI")){
112     area = new SquareMili(areaValue);
113     switch (unit2){
114         case "SF":
115             System.out.println("\n" + areaValue + " SquareMili in SquareF: " + String.format("%.2f", area.toSquareF()));
116             break;
117         case "SM":
118             System.out.println("\n" + areaValue + " SquareMili in SquareM: " + String.format("%.2f", area.toSquareM()));
119             break;
120         case "SI":
121             System.out.println("\n" + areaValue + " SquareMili in SquareI: " + String.format("%.2f", area.toSquareI()));
122             break;
123         case "SK":
124             System.out.println("\n" + areaValue + " SquareMili in SquareKM: " + String.format("%.2f", area.toSquareKM()));
125             break;
126         case "SMLI":
127             System.out.println("\n" + areaValue + " SquareMili in SquareMili: " + String.format("%.2f", area.toSquareMili()));
128             break;
129         case "SCM":
130             System.out.println("\n" + areaValue + " SquareMili in SquareCM: " + String.format("%.2f", area.toSquareCM()));
131             break;
132     }
133 }
134 }else if (unit.equalsIgnoreCase( anotherString: "SCM")){
135     area = new SquareCM(areaValue);
136     switch (unit2){
137         case "SF":
138             System.out.println("\n" + areaValue + " SquareCM in SquareF: " + String.format("%.2f", area.toSquareF()));
139             break;
140         case "SM":
141             System.out.println("\n" + areaValue + " SquareCM in SquareM: " + String.format("%.2f", area.toSquareM()));
142             break;
143         case "SI":
144             System.out.println("\n" + areaValue + " SquareCM in SquareI: " + String.format("%.2f", area.toSquareI()));
145             break;
146         case "SK":
147             System.out.println("\n" + areaValue + " SquareCM in SquareKM: " + String.format("%.2f", area.toSquareKM()));
148             break;
149         case "SMLI":
150             System.out.println("\n" + areaValue + " SquareCM in SquareMili: " + String.format("%.2f", area.toSquareMili()));
151             break;
152         case "SCM":
153             System.out.println("\n" + areaValue + " SquareCM in SquareCM: " + String.format("%.2f", area.toSquareCM()));
154             break;
155     }
156 }else{
157     System.out.print("Invalid area unit");
158     return;
159 }
160 }
}

```

Area class

```
abstract class Area {  
    37 usages  
    protected double value;  
  
    6 usages  
    public Area(double value) { this.value = value; }  
  
    6 usages 6 implementations  
    public abstract double toSquareM();  
  
    6 usages 6 implementations  
    public abstract double toSquareF();  
  
    6 usages 6 implementations  
    public abstract double toSquareI();  
    6 usages 6 implementations  
    public abstract double toSquareKM();  
    6 usages 6 implementations  
    public abstract double toSquareMili();  
  
    6 usages 6 implementations  
    public abstract double toSquareCM();  
  
}  
1 usage  
class SquareM extends Area{  
    1 usage  
    public SquareM(double value) { super(value); }  
    6 usages  
    public double toSquareM() { return value; }  
    6 usages  
    public double toSquareF() { return value * 10.76391; }  
    6 usages  
    public double toSquareI() { return value * 15504; }  
    6 usages  
    public double toSquareKM(){return value * (10*-6);}  
    6 usages  
    public double toSquareMili() { return value * 1000000; }  
    6 usages  
    public double toSquareCM() { return value * 10000; }  
}  
1 usage
```

```
class SquareF extends Area{
    1 usage
    public SquareF(double value) { super(value); }
    6 usages
    public double toSquareF() { return value; }
    6 usages
    public double toSquareM() { return value * 0.09290304; }
    6 usages
    public double toSquareI() { return value * 144; }
    6 usages
    public double toSquareKM(){return value * 0.000000092903;}
    6 usages
    public double toSquareMili() { return value * 92903; }
    6 usages
    public double toSquareCM() { return value * 929.03; }
}

1 usage
class SquareI extends Area{
    1 usage
    public SquareI(double value) { super(value); }
    6 usages
    public double toSquareI() { return value; }
    6 usages
    public double toSquareF() { return value * 0.00694444; }
    6 usages
    public double toSquareM() { return value * 0.00064516; }
    6 usages
    public double toSquareKM(){return value * 0.0000000064516;}
    6 usages
    public double toSquareMili() { return value * 645.16; }
    6 usages
    public double toSquareCM() { return value * 6.4516; }
}
```

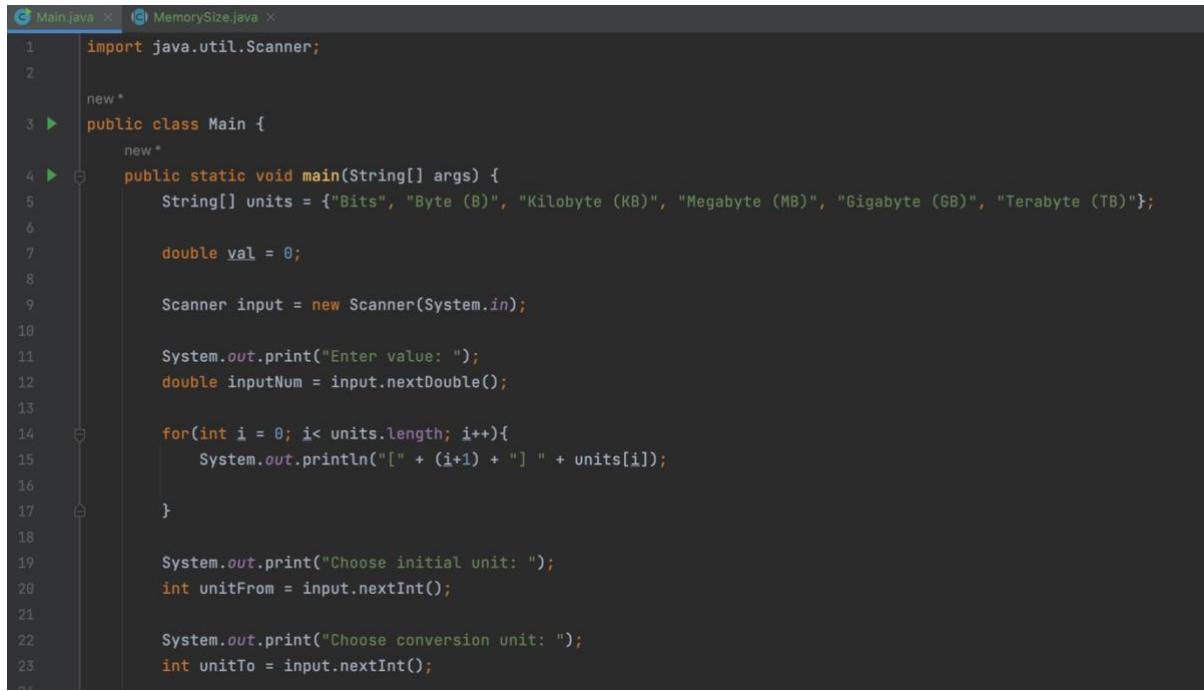
```
class SquareKM extends Area{
    1 usage
    public SquareKM(double value) { super(value); }
    6 usages
    public double toSquareKM() { return value; }
    6 usages
    public double toSquareF() { return value * 0.00001076; }
    6 usages
    public double toSquareI() { return value * 1550003100; }
    6 usages
    public double toSquareM(){return value * 1000000;}
    6 usages
    public double toSquareMili() { return value * 0.00000000001; }
    6 usages
    public double toSquareCM() { return value * (10*10); }
}

1 usage
class SquareMili extends Area{
    1 usage
    public SquareMili(double value){super(value);}
    6 usages
    public double toSquareMili() { return value; }
    6 usages
    public double toSquareF() { return value * 0.000010764; }
    6 usages
    public double toSquareI() { return value * 0.00155; }
    6 usages
    public double toSquareKM(){return value * (10*12);}
    6 usages
    public double toSquareM() { return value * 0.000001; }
    6 usages
    public double toSquareCM() { return value * 0.01; }
}
```

```
class SquareCM extends Area{
    1 usage
    public SquareCM(double value) { super(value); }
    6 usages
    public double toSquareCM() { return value; }
    6 usages
    public double toSquareF() { return value * 0.0010763888889; }
    6 usages
    public double toSquareI() { return value * 0.155; }
    6 usages
    public double toSquareKM(){return value * (10*10);}
    6 usages
    public double toSquareMili() { return value * 100; }
    6 usages
    public double toSquareM() { return value * 0.0001; }
}
```

Memory Size

Main class



```
1 import java.util.Scanner;
2
3 public class Main {
4     new *
5     public static void main(String[] args) {
6         String[] units = {"Bits", "Byte (B)", "Kilobyte (KB)", "Megabyte (MB)", "Gigabyte (GB)", "Terabyte (TB)"};
7
8         double val = 0;
9
10        Scanner input = new Scanner(System.in);
11
12        System.out.print("Enter value: ");
13        double inputNum = input.nextDouble();
14
15        for(int i = 0; i< units.length; i++){
16            System.out.println("[ " + (i+1) + " ] " + units[i]);
17        }
18
19        System.out.print("Choose initial unit: ");
20        int unitFrom = input.nextInt();
21
22        System.out.print("Choose conversion unit: ");
23        int unitTo = input.nextInt();
```

```
>Main.java × MemorySize.java ×
25     if(unitFrom == 1){
26         Bits bt = new Bits(inputNum);
27
28         // double num = 0;
29
30         switch (unitTo) {
31             case 1:
32                 val = bt.toBits();
33                 break;
34             case 2:
35                 val = bt.toByte();
36                 break;
37             case 3:
38                 val = bt.toKb();
39                 break;
40             case 4:
41                 val = bt.toMb();
42                 break;
43             case 5:
44                 val = bt.toGb();
45                 break;
46             case 6:
47                 val = bt.toTb();
48                 break;
49             default:{
50
51             }
52         }
53
54         System.out.println(val);
55     }
56
57     else if(unitFrom == 2){
58         Byte byt = new Byte(inputNum);
59
60         switch (unitTo){
```

```
>Main.java  ×  MemorySize.java  ×
50
51     } else if(unitFrom == 2){
52         Byte byt = new Byte(inputNum);
53
54         switch (unitTo){
55             case 1:
56                 val = byt.toBits();
57                 break;
58             case 2:
59                 val = byt.toByte();
60                 break;
61             case 3:
62                 val = byt.toKb();
63                 break;
64             case 4:
65                 val = byt.toMb();
66                 break;
67             case 5:
68                 val = byt.toGb();
69                 break;
70             case 6:
71                 val = byt.toTb();
72                 break;
73             default:{
74
75                 }
76             }
77
78         }
79
80         System.out.println(val);
81     }
82
83
84     } else if(unitFrom == 3){
85         KiloBytes kb = new KiloBytes(inputNum);
86
87         switch (unitTo) {
88             case 1:
89                 val = kb.toBits();
90             }
91
92         }
93
94     }
95 }
```

```
>Main.java  ×  MemorySize.java  ×
87     } else if(unitFrom == 3){
88         KiloBytes kb = new KiloBytes(inputNum);
89
90         switch (unitTo) {
91             case 1:
92                 val = kb.toBits();
93                 break;
94             case 2:
95                 val = kb.toByte();
96                 break;
97             case 3:
98                 val = kb.toKb();
99                 break;
100            case 4:
101                val = kb.toMb();
102                break;
103            case 5:
104                val = kb.toGb();
105                break;
106            case 6:
107                val = kb.toTb();
108                break;
109            default: {
110
111            }
112        }
113
114        System.out.println(val);
115    }
116
117    else if(unitFrom == 4){
118        MegaByte mb = new MegaByte(inputNum);
119
120        switch (unitTo) {
121            case 1:
122                val = mb.toBits();
```

```
>Main.java  ×  MemorySize.java  ×
117
118     else if(unitFrom == 4){
119         MegaByte mb = new MegaByte(inputNum);
120
121         switch (unitTo) {
122             case 1:
123                 val = mb.toBits();
124                 break;
125             case 2:
126                 val = mb.toByte();
127                 break;
128             case 3:
129                 val = mb.toKb();
130                 break;
131             case 4:
132                 val = mb.toMb();
133                 break;
134             case 5:
135                 val = mb.toGb();
136                 break;
137             case 6:
138                 val = mb.toTb();
139                 break;
140             default: {
141                 }
142             }
143
144             System.out.println(val);
145         }
146
147     else if(unitFrom == 5){
148         GigaByte gb = new GigaByte(inputNum);
149
150         switch (unitTo) {
151             case 1:
152                 gb.toBits();
```

```
147     else if(unitFrom == 5){
148         GigaByte gb = new GigaByte(inputNum);
149
150         switch (unitTo) {
151             case 1:
152                 gb.toBits();
153                 break;
154             case 2:
155                 gb.toByte();
156                 break;
157             case 3:
158                 gb.toKb();
159                 break;
160             case 4:
161                 gb.toMb();
162                 break;
163             case 5:
164                 gb.toGb();
165                 break;
166             case 6:
167                 gb.toTb();
168                 break;
169             default: {
170
171                 }
172             }
173
174             System.out.println(val);
175         }
176
177     else if(unitFrom == 6){
178         TeraByte tb = new TeraByte(inputNum);
179
180         switch (unitTo) {
181             case 1:
182                 val = tb.toBits();
```

```
176  
177     }  
178     else if(unitFrom == 6){  
179         TeraByte tb = new TeraByte(inputNum);  
180  
181         switch (unitTo) {  
182             case 1:  
183                 val = tb.toBits();  
184                 break;  
185             case 2:  
186                 val = tb.toByte();  
187                 break;  
188             case 3:  
189                 val = tb.toKb();  
190                 break;  
191             case 4:  
192                 val = tb.toMb();  
193                 break;  
194             case 5:  
195                 val = tb.toGb();  
196                 break;  
197             case 6:  
198                 val = tb.toTb();  
199                 break;  
200             default: {  
201                 }  
202             }  
203             System.out.println(val);  
204         }  
205     }  
206     }  
207 }  
208 }  
209 }
```

Memory Size class

The screenshot shows a Java code editor with two tabs: `Main.java` and `MemorySize.java`. The `MemorySize.java` tab is active, displaying the following code:

```
1 6 inheritors new *
2 public abstract class MemorySize {
3     protected double value;
4
5     static double kilo = 1024;
6     static double mega = kilo * kilo;
7     static double giga = mega * kilo;
8     static double tera = giga * kilo;
9
10    new *
11    public MemorySize(double value) { this.value = value; }
12
13    6 implementations new *
14    abstract double toBits();
15    6 implementations new *
16    abstract double toByte();
17    6 implementations new *
18    abstract double toKb();
19    6 implementations new *
20    abstract double toMb();
21    6 implementations new *
22    abstract double toGb();
23    6 implementations new *
24    abstract double toTb();
25 }
```

The `Bits` class is defined in the `Main.java` tab:

```
new *
24 class Bits extends MemorySize{
25     new *
26     public Bits(double value) { super(value); }
27
28     new *
29     public double toBits() { return value; }
30
31     new *
32     public double toByte() { return value / 8; }
33
34     new *
35     public double toKb(){
36         new *
37         public double toKb(){
38             new *
39             return (value / 8) / kilo;
40         }
41
42     new *
43     public double toMb(){
44         new *
45         return (value / 8) / mega;
46     }
47
48     new *
49     public double toGb(){
50         new *
51         return (value / 8) / giga;
52     }
53
54     new *
55     public double toTb() {
56         new *
57             return (value / 8) / tera;
58     }
59 }
```

```
>Main.java x | MemorySize.java x
new *
59 class Byte extends MemorySize{
    new *
60     public Byte(double value) { super(value); }
63
    new *
64     public double toBits(){
65         return value * 8;
66     }
67
    new *
68     public double toByte(){
69         return value;
70     }
71
    new *
72     public double toKb(){
73         return value / kilo;
74     }
75
    new *
76     public double toMb(){
77         return value / mega;
78     }
79
    new *
80     public double toGb(){
81         return value / giga;
82     }
83
    new *
84     public double toTb(){
85         return value / tera;
86     }
87 }
```

```
>Main.java <-- MemorySize.java <--  
89  class KiloBytes extends MemorySize{  
90  
91      new *  
92      public KiloBytes(double value) { super(value); }  
94  
95  ⏷  new *  
96      public double toBits(){  
97          return value * mega * 8;  
98      }  
99  ⏷  new *  
100     public double toByte(){  
101         return value * kilo;  
102     }  
103  ⏷  new *  
104      public double toKb(){  
105          return value;  
106      }  
107  ⏷  new *  
108      public double toMb(){  
109          return value / kilo;  
110      }  
111  ⏷  new *  
112      public double toGb(){  
113          return value / mega;  
114      }  
115  ⏷  new *  
116      public double toTb(){  
117          return value / giga;  
118      }  
119 }
```

```
>Main.java <-- MemorySize.java <--  
120     class MegaByte extends MemorySize{  
121         new *  
122         public MegaByte(double value) { super(value); }  
124  
125     new *  
126     public double toBits(){  
127         return value / mega * 8;  
128     }  
129  
130     new *  
131     public double toByte(){  
132         return value * mega;  
133     }  
134     new *  
135     public double toKb(){  
136         return value * kilo;  
137     }  
138     new *  
139     public double toMb(){  
140         return value;  
141     }  
142     new *  
143     public double toGb(){  
144         return value / kilo;  
145     }  
146     new *  
147     public double toTb(){  
148         return value / mega;  
149     }
```

```
>Main.java <-- MemorySize.java <--  
152     class GigaByte extends MemorySize{  
153  
154         new *  
155             public GigaByte(double value) { super(value); }  
156  
157         new *  
158         public double toBits(){  
159             return value * giga * 8;  
160         }  
161  
162         new *  
163         public double toByte(){  
164             return value * giga;  
165         }  
166         new *  
167         public double toKb(){  
168             return value * mega;  
169         }  
170         new *  
171         public double toMb(){  
172             return value * kilo;  
173         }  
174         new *  
175         public double toGb(){  
176             return value;  
177         }  
178         new *  
179         public double toTb(){  
180             return value / kilo;  
181         }  
182 }
```

```
>Main.java <--> MemorySize.java
```

```
183     class TeraByte extends MemorySize{
184         new *
185             public TeraByte(double value) { super(value); }
186
187             new *
188             public double toBits(){
189                 return value * tera * 8;
190             }
191
192             new *
193             public double toByte(){
194                 return value * tera;
195             }
196
197             new *
198             public double toKb(){
199                 return value * giga;
200             }
201
202             new *
203             public double toMb(){
204                 return value * mega;
205             }
206
207             new *
208             public double toGb(){
209                 return value * kilo;
210             }
211
212             new *
213             public double toTb(){
214                 return value;
215             }
216 }
```

Category Class

-used to organize the options chosen by the user.

Main Function (Test Code for Category)

```
27 ►   public static void main(String[] args) {  
28     Scanner in = new Scanner(System.in);  
29     System.out.println("WELCOME TO UNICON");  
30  
31     System.out.println("CATEGORIES:");  
32     System.out.println("\tAREA");  
33     System.out.println("\tANGLE");  
34     System.out.println("\tBITS");  
35     System.out.println("\tCURRENCY");  
36     System.out.println("\tLENGTH");  
37     System.out.println("\tMASS");  
38     System.out.println("\tTIME");  
39     System.out.println("\tTEMPERATURE");  
40     System.out.println("\tPRESSURE");  
41     System.out.println("\tVOLUME");  
42     System.out.print("\nChoose a category: ");  
43     String choice = in.next().toUpperCase();  
44  
45     Category.chooseCategory(choice);  
46  
47 }
```

Main Function (Test Code for Category)

```
Category.java ×  
1 import java.util.Scanner;  
2  
3 public class Category {  
4  
5     //list of categories available  
6     private final String[] category = {"AREA", "ANGLE", "BITS", "CURRENCY", "LENGTH", "MASS",  
7                                         "TIME", "TEMPERATURE", "PRESSURE", "VOLUME"};  
8  
9     //choose conversion category based on user choice  
10    @  
11    public static void chooseCategory(String category){  
12        switch (category) {  
13            case "AREA" -> Area.convert();  
14            case "ANGLE" -> Angle.convert();  
15            case "BITS" -> Bits.convert();  
16            case "CURRENCY" -> Currency.convert();  
17            case "LENGTH" -> Length.convert();  
18            case "MASS" -> Mass.convert();  
19            case "TIME" -> Time.convert();  
20            case "Temperature" -> Temperature.convert();  
21            case "PRESSURE" -> Pressure.convert();  
22            case "VOLUME" -> Volume.convert();  
23        }  
24    }
```