



UNIVERSITI
TEKNOLOGI
PETRONAS

TEB1043 Object Oriented Programming

January 2023 Semester

Department of Computer and Information Sciences

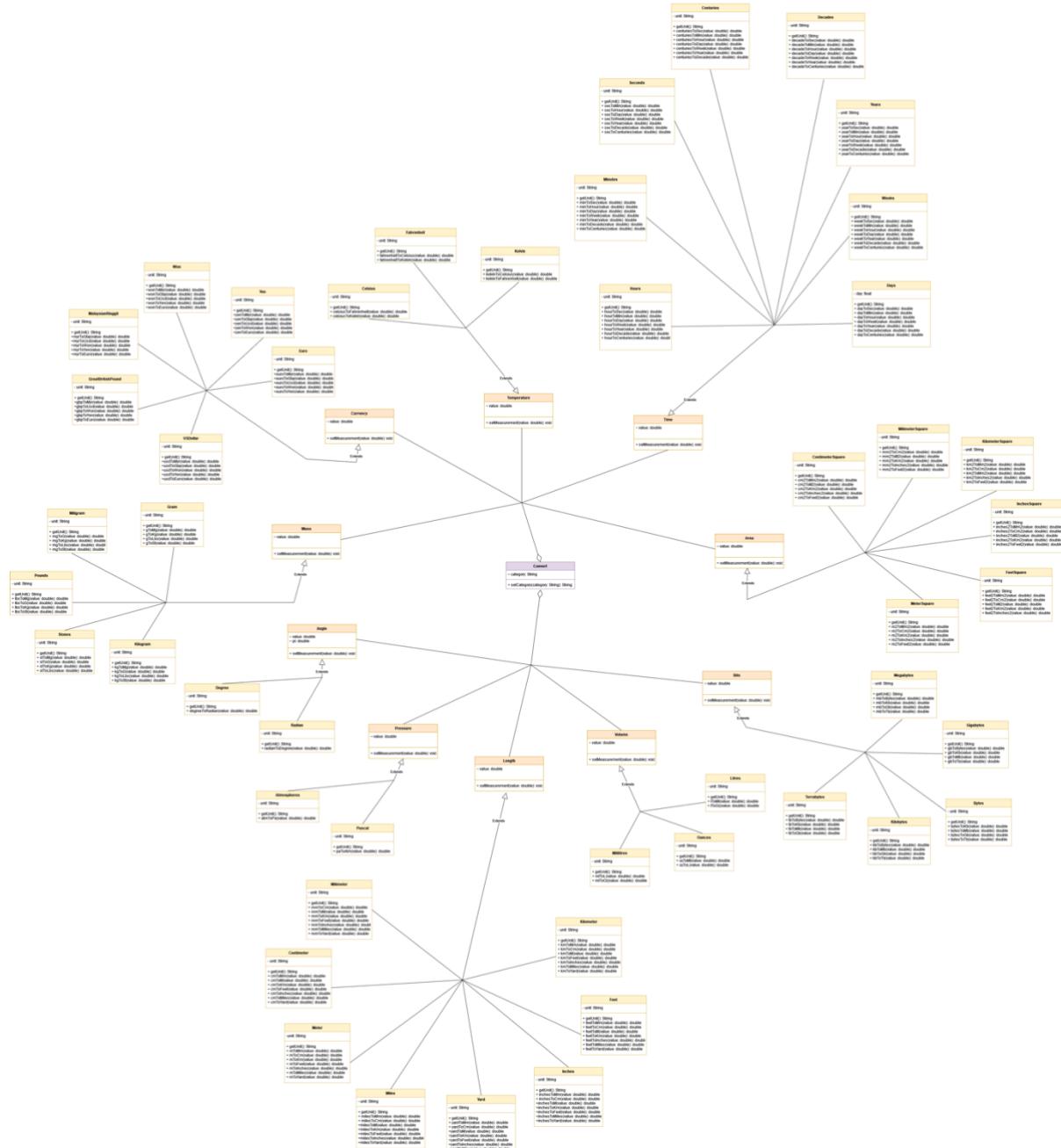
Prepared for:

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	Muhammad Izhar
Currency	Iman Kamil
Angle	Ammar Afif
Pressure	Ikhwan Hanafi
Area	Danish Hariz
Volume	Ahmad Zafri
Bits	Aqil Fauzan

Source Codes

Temperature Class

```
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 Fahrenheit(tempValue); // Instantiates a Fahrenheit 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 X J Temperature.java
src > J TempConverter.java > ↗ TempConverter > ⚡ main(String[])
35
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 X J Temperature.java

```
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
```

Angle Class

The screenshot shows two Java files open in a code editor:

Main.java

```
1 import java.util.Scanner;
2
3 public class Main {
4
5     public static void main(String[] args) {
6
7         Scanner in = new Scanner(System.in);
8
9         System.out.println("\n---Choose angle unit to convert from---");
10        System.out.println("Degree (D)");
11        System.out.println("Radian (R)");
12        System.out.print("Choice (D/R): ");
13        String unit = in.nextLine().toUpperCase();
14
15        System.out.print("\nEnter value of angle: ");
16        double meas = in.nextDouble();
17
18        if (Unit.equalsIgnoreCase(anotherString: "D")){
19            Degree deg = new Degree();
20            System.out.println("\n" + meas + " " + deg.getUnit() + " in radian: " + String.format("%.4f", deg.degreeToRadian(meas)));
21        } else if (unit.equalsIgnoreCase(anotherString: "R")) {
22            Radian rad = new Radian();
23            System.out.println("\n" + meas + " " + rad.getUnit() + " in degree: " + String.format("%.4f", rad.radianToDegree(meas)));
24        } else{
25            System.out.println("Invalid unit!");
26            return;
27        }
28
29        in.close();
30    }
31
32 }
```

Angle.java

```
1 2 inheritors
2
3 public class Angle {
4
5     protected double measurement;
6     protected final double pi = Math.PI;
7
8     public void setMeasurement(double measurement) { this.measurement = measurement; }
9
10 }
11
12 class Degree extends Angle{
13     private String unit = "Degree";
14
15     public String getUnit() { return this.unit; }
16
17     public double degreeToRadian(double measurement){
18         double ans;
19
20         ans = measurement * pi / 180;
21
22         return ans;
23     }
24
25 }
26
27
28 class Radian extends Angle{
29     private String unit = "Radian";
30
31     public String getUnit(){
32         return this.unit;
33     }
34
35     public double radianToDegree(double measurement){
36         double ans;
37
38         ans = measurement * 180 / pi;
39
40         return ans;
41     }
42
43 }
```

The screenshot shows two Java files open in a code editor:

Main.java

```
1 Main.java × 2 Angle.java ×
2
3
4
```

Angle.java

```
1 2 inheritors
2
3 public class Angle {
4
5     protected double measurement;
6     protected final double pi = Math.PI;
7
8     public void setMeasurement(double measurement) { this.measurement = measurement; }
9
10 }
11
12 class Degree extends Angle{
13     private String unit = "Degree";
14
15     public String getUnit() { return this.unit; }
16
17     public double degreeToRadian(double measurement){
18         double ans;
19
20         ans = measurement * pi / 180;
21
22         return ans;
23     }
24
25 }
26
27
28 class Radian extends Angle{
29     private String unit = "Radian";
30
31     public String getUnit(){
32         return this.unit;
33     }
34
35     public double radianToDegree(double measurement){
36         double ans;
37
38         ans = measurement * 180 / pi;
39
40         return ans;
41     }
42
43 }
```

Length Class

```
>Main.java x | Length.java x
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/M/MI/YD/IN/FT/KM): ");
16        String unit = input.next();
17        input.close();
18
19        Length length;
20        if (unit.equalsIgnoreCase( anotherString: "MM")){
21            length = new Milimeter(lengthValue);
22        }else if (unit.equalsIgnoreCase( anotherString: "CM")){
23            length = new Centimeter(lengthValue);
24        }else if (unit.equalsIgnoreCase( anotherString: "M")){
25            length = new Meter(lengthValue);
26        }else if (unit.equalsIgnoreCase( anotherString: "MI")){
27            length = new Miles(lengthValue);
28        }else if (unit.equalsIgnoreCase( anotherString: "YD")){
29            length = new Yard(lengthValue);
30        }else if (unit.equalsIgnoreCase( anotherString: "IN")){
31            length = new Inches(lengthValue);
32        }else if (unit.equalsIgnoreCase( anotherString: "FT")){
33            length = new Feet(lengthValue);
34        }else if (unit.equalsIgnoreCase( anotherString: "KM")){
35            length = new Kilometer(lengthValue);
36        }else{
37            System.out.print("Invalid length unit");
38            return;
39        }
40        System.out.println("Length in Milimeter: " + formatter.format(length.toMilimeter()));
41        System.out.println("Length in Centimeter: " + formatter.format(length.toCentimeter()));
42        System.out.println("Length in Meter: " + formatter.format(length.toMeter()));
43        System.out.println("Length in Miles: " + formatter.format(length.toMiles()));
44        System.out.println("Length in Yard: " + formatter.format(length.toYard()));
45        System.out.println("Length in Inches: " + formatter.format(length.toInches()));
46        System.out.println("Length in Feet: " + formatter.format(length.toFeet()));
47        System.out.println("Length in Kilometer: " + formatter.format(length.toKilometer()));
48
49    }
50 }
```

The image shows two side-by-side code editors, both titled "Length.java".

Left Editor (Line 1 to 21):

```
1  o↓ | abstract class Length {  
2      9 usages 8 inheritors  
3      65 usages  
4      protected double value;  
5  
6      8 usages  
7      public Length(double value) {  
8          this.value = value;  
9      }  
10     1 usage 8 implementations  
11     public abstract double toMilimeter();  
12     1 usage 8 implementations  
13     public abstract double toCentimeter();  
14     1 usage 8 implementations  
15     public abstract double toMeter();  
16     1 usage 8 implementations  
17     public abstract double toYard();  
18     1 usage 8 implementations  
19     public abstract double toInches();  
20     1 usage 8 implementations  
21     public abstract double toFeet();
```

Right Editor (Line 22 to 44):

```
22  o↓ | 1 usage 8 implementations  
23  
24  } 1 usage  
25  class Milimeter extends Length{  
26      1 usage  
27      public Milimeter(double value){  
28          super(value);  
29  } 1 usage  
30      public double toMilimeter(){  
31          return value;  
32  } 1 usage  
33      public double toCentimeter(){  
34          return value / 10.0;  
35  
36  o↑ | 1 usage  
37      public double toMeter(){  
38          return value / 1000.0;  
39  
40  o↑ | 1 usage  
41      public double toYard(){  
42          return value / 1609344.0;  
43  o↑ | 1 usage  
44      public double toInches(){  
45          return value / 914.4;
```

```
Main.java x Length.java x
47 ①↑    public double toInches(){
48      return value / 25.4;
49  }
50 ①↑    public double toFeet(){
51      return value / 304.8;
52  }
53 ①↑    public double toKilometer(){
54      return value / 1000000.0;
55  }
56 ①↑    }
57
58     class Centimeter extends Length{
59         public Centimeter(double value){
60             super(value);
61         }
62 ①↑    public double toMilimeter(){
63      return value * 10.0;
64  }
65 ①↑    public double toCentimeter(){
66      return value;
67  }
68
69 ①↑    public double toMeter() {
70      return value / 100.0;
71  }
72
73 ①↑    public double toMiles(){
74      return value / 160934.4;
75  }
76 ①↑    public double toYard(){
77      return value / 91.44;
78  }
79 ①↑    public double toInches(){
80      return value / 2.54;
81  }
82 ①↑    public double toFeet(){
83      return value / 30.48;
84  }
85 ①↑    public double toKilometer(){
86      return value / 100000.0;
87  }
88 ①↑    }
89
90     class Meter extends Length {
91         public Meter(double value) {
92             super(value);
93         }
94
95 ①↑    public double toMilimeter() {
96      return value * 1000.0;
97  }
```

```
>Main.java × | Length.java × | Main.java × | Length.java × |
99 ⚡ 1 usage
100     public double toCentimeter() {
101         return value * 100.0;
102     }
103 ⚡ 1 usage
104     public double toMeter() {
105         return value;
106     }
107 ⚡ 1 usage
108     public double toMiles() {
109         return value / 1609.0;
110     }
111 ⚡ 1 usage
112     public double toYard() {
113         return value * 1.094;
114     }
115 ⚡ 1 usage
116     public double toInches() {
117         return value * 39.37;
118     }
119 ⚡ 1 usage
120     public double toFeet() {
121         return value * 3.281;
123 ⚡ 1 usage
124     public double toKilometer() {
125         return value / 1000.0;
126     }
127 class Miles extends Length {
128     public Miles(double value) {
129         super(value);
130     }
131 ⚡ 1 usage
132     public double toMilimeter() {
133         return value * 1609344;
134 ⚡ 1 usage
135     public double toCentimeter() {
136         return value * 160934.4;
137     }
138 ⚡ 1 usage
139     public double toMeter() {
140         return value * 1609.344;
141     }
142 ⚡ 1 usage
143     public double toMiles() {
144         return value;
145 }
```

```
Main.java
146     public double toYard() {
147         return value * 1760;
148     }
149
150     public double toInches() {
151         return value * 63360;
152     }
153
154     public double toFeet() {
155         return value * 5280;
156     }
157
158     public double toKilometer() {
159         return value * 1.609344;
160     }
161
162     class Yard extends Length {
163         public Yard(double value) {
164             super(value);
165         }
166
167         public double toMilimeter() {
168             return value * 914.4;
169         }
170     }
171
172     public double toCentimeter() {
173         return value * 91.44;
174     }
175
176     public double toMeter() {
177         return value * 0.9144;
178     }
179
180     public double toMiles() {
181         return value / 1760;
182     }
183
184     public double toYard() {
185         return value;
186     }
187
188     public double toInches() {
189         return value * 36;
190     }
191
192     public double toFeet() {
193         return value * 3;
194     }
195
196     public double toKilometer() {
```

```
>Main.java x Length.java x
195 ⚡  public double toKilometer() {
196      return value * 0.0009144;
197  }
198  }
199  class Inches extends Length {
200      public Inches(double value) {
201          super(value);
202      }
203      public double toMilimeter() {
204          return value * 25.4;
205      }
206  }
207  public double toCentimeter() {
208      return value * 2.54;
209  }
210  public double toMeter() {
211      return value * 0.0254;
212  }
213  public double toMiles() {
214      return value / 63360;
215  }
216  public double toYard() {
217      return value / 36;
218  }
219  class Feet extends Length {
220      public Feet(double value) {
221          super(value);
222      }
223      public double toInches() {
224          return value;
225      }
226      public double toFeet() {
227          return value / 12;
228      }
229      public double toKilometer() {
230          return value * 0.0000254;
231      }
232      public double toMilimeter() {
233          return value * 304.8;
234  }
235  }
236  class Feet extends Length {
237      public Feet(double value) {
238          super(value);
239      }
240      public double toInches() {
241          return value * 304.8;
242  }
243  }
```

```
>Main.java x Length.java x
246 ⚡  public double toCentimeter() {
247     return value * 30.48;
248 }
249
250 ⚡  1 usage
251  public double toMeter() {
252     return value * 0.3048;
253 }
254 ⚡  1 usage
255  public double toMiles() {
256     return value / 5280;
257 }
258 ⚡  1 usage
259  public double toYard() {
260     return value / 3;
261 }
262 ⚡  1 usage
263  public double toInches() {
264     return value * 12;
265 }
266 ⚡  1 usage
267  public double toFeet() {
268     return value;
269 }
270 ⚡  1 usage
271  public double toKilometer() {
```

```
>Main.java x Length.java x
270 ⚡  public double toKilometer() {
271     return value / 3280.84;
272 }
273 }
274
275 1 usage
276  class Kilometer extends Length {
277
278     public Kilometer(double value) {
279         super(value);
280
281     }
282
283
284 ⚡  1 usage
285  public double toMilimeter() {
286     return value * 1000000;
287 }
288 ⚡  1 usage
289  public double toCentimeter() {
290     return value * 100000;
291 }
292 ⚡  1 usage
293  public double toMiles() {
294     return value / 1.609344;
```

```
Main.java x Length.java x
1 usage
292 ①↑ 1 usage
293     public double toMiles() {
294         return value / 1.609344;
295     }
296 ①↑ 1 usage
297     public double toYard() {
298         return value * 1093.61;
299     }
300 ①↑ 1 usage
301     public double toInches() {
302         return value * 39370.1;
303     }
304 ①↑ 1 usage
305     public double toFeet() {
306         return value * 3280.84;
307     }
308 ①↑ 1 usage
309     public double toKilometer() {
310         return value;
311     }
312 }
```

Time Class

```
Main.java
import java.text.DecimalFormat;
import java.text.NumberFormat;
import java.util.*;
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("##.##");

        System.out.print("Enter time value: ");
        double timeValue = input.nextDouble();

        System.out.print("Enter time unit (S/Min/H/D/W/M/Y/Dec/C): ");
        String unit = input.next();
        input.close();

        // Creates an instance of the class
        Time duration;
        if (unit.equalsIgnoreCase("S")) {
            duration = new Second(timeValue);
        }
        else if (unit.equalsIgnoreCase("Min")){
            duration = new Minute(timeValue);
        }
        else if (unit.equalsIgnoreCase("H")){
            duration = new Hour(timeValue);
        }
        else if (unit.equalsIgnoreCase("D")){
            duration = new Day(timeValue);
        }
        else if (unit.equalsIgnoreCase("W")){
            duration = new Week(timeValue);
        }
        else if (unit.equalsIgnoreCase("M")){
            duration = new Month(timeValue);
        }
        else if (unit.equalsIgnoreCase("Y")){
            duration = new Year(timeValue);
        }
        else if (unit.equalsIgnoreCase("Dec")){
            duration = new Decade(timeValue);
        }
        else if (unit.equalsIgnoreCase("C")){
            duration = new Century(timeValue);
        }
        else{
            System.out.print("Invalid time unit!");
            return;
        }

        System.out.println("Time in Second: " + formatter.format(duration.toSecond()));
        System.out.println("Time in Minute: " + formatter.format(duration.toMin()));
        System.out.println("Time in Hour: " + formatter.format(duration.toHour()));
        System.out.println("Time in Day: " + formatter.format(duration.toDay()));
        System.out.println("Time in Week: " + formatter.format(duration.toWeek()));
        System.out.println("Time in Month: " + formatter.format(duration.toMonth()));
        System.out.println("Time in Year: " + formatter.format(duration.toYear()));
        System.out.println("Time in Decade: " + formatter.format(duration.toDecade()));
        System.out.println("Time in Century: " + formatter.format(duration.toCentury()));
    }
}

Time.java
class Time {
    private double value;
    private String unit;

    public Time(double value) {
        this.value = value;
        this.unit = null;
    }

    public Time(double value, String unit) {
        this.value = value;
        this.unit = unit;
    }

    public double toSecond() {
        return value;
    }

    public double toMin() {
        return value / 60;
    }

    public double toHour() {
        return value / 3600;
    }

    public double toDay() {
        return value / 86400;
    }

    public double toWeek() {
        return value / 604800;
    }

    public double toMonth() {
        return value / 2628000;
    }

    public double toYear() {
        return value / 31536000;
    }

    public double toDecade() {
        return value / 315360000;
    }

    public double toCentury() {
        return value / 31536000000;
    }

    public String toString() {
        return value + " " + unit;
    }
}
```

```
62     class Second extends Time {           87 ⚡|    public double toYear() {  
63         1 usage                           88     return value/31536000;  
64         public Second(double value) {      89     }  
65             super(value);  
66         }  
67     ⚡|    1 usage  
68         public double toSecond() {          90 ⚡|    public double toDecade() {  
69             return value;                  91     return value/315360000;  
70         }  
71     ⚡|    1 usage  
72         public double toMin() {            92     }  
73             return (value/60);  
74         }  
75     ⚡|    1 usage  
76         public double toHour() {           93 ⚡|    public double toCentury() {  
77             return (value/3600);  
78     ⚡|    1 usage  
79         public double toDay() {            94     return value/315360000;  
80             return (value/86400);  
81     ⚡|    1 usage  
82         public double toWeek() {           95     }  
83             return value/604800;  
84     ⚡|    1 usage  
85         public double toMonth() {          96     }  
86             return value/2628000;  
87     ⚡|    1 usage  
88         public double toYear() {  
89             return value/31536000;  
90     ⚡|    1 usage  
91         public double toDecade() {  
92             return value/315360000;  
93     ⚡|    1 usage  
94         public double toCentury() {  
95             return value/315360000;  
96     ⚡|    1 usage  
97         class Minute extends Time {  
98             1 usage  
99             public Minute(double value) {  
100                super(value);  
101            }  
102    ⚡|    1 usage  
103        public double toSecond() {  
104            return value*60;  
105        }  
106    ⚡|    1 usage  
107        public double toMin() {  
108            return value;  
109        }  
110    ⚡|    1 usage  
111        public double toHour() {  
112            return (value/60);
```

```
Main.java
113 ①↑    public double toDay() {
114      return value/1440;
115  }
116 ①↑    public double toWeek() {
117      return value/10080;
118  }
119 ①↑    public double toMonth() {
120      return value/43829.1;
121  }
122 ①↑    public double toYear() {
123      return value/525949;
124  }
125 ①↑    public double toDecade() {
126      return value/5259492;
127  }
128 ①↑    public double toCentury() {
129      return value/52596000;
130  }
131
132  }
133  class Hour extends Time {
134      public Hour(double value) {
135          super(value);
136  }

Time.java
138 ①↑    public double toSecond() {
139      return value*3600;
140  }
141
142 ①↑    public double toMin() {
143      return value*60;
144  }
145
146 ①↑    public double toHour() {
147      return value;
148  }
149 ①↑    public double toDay() {
150      return value/24;
151  }
152 ①↑    public double toWeek() {
153      return value/168;
154  }
155 ①↑    public double toMonth() {
156      return value/730.484;
157  }
158 ①↑    public double toYear() {
159      return value/8760;
160  }
161 ①↑    public double toDecade() {
162      return value/87600;
163  }
```

```
Main.java × Time.java × Main.java × Time.java ×
164 1 usage
165     public double toCentury() {
166         return value/876000;
167     }
168 }
169 class Day extends Time {
170     1 usage
171     public Day(double value) {
172         super(value);
173     }
174     1 usage
175     public double toSecond() {
176         return value*86400;
177     }
178     1 usage
179     public double toMin() {
180         return (value*1440);
181     }
182     1 usage
183     public double toHour() {
184         return (value*24);
185     }
186     1 usage
187     public double toDay() {
188     }
189
190
191 1 usage
192     public double toWeek() {
193         return value/7;
194     }
195
196
197 1 usage
198     public double toMonth() {
199         return value/30;
200     }
201
202
203 1 usage
204     class Week extends Time {
205         1 usage
206         public Week(double value) {
207             super(value);
208         }
209     }
210
211
212
```

```
Main.java x Time.java x
  usage
213 ①↑    public double toMin() {
214      return value*10080;
215  }
216
217 ①↑    public double toHour() {
218      return value*168;
219  }
220 ①↑    public double toDay() {
221      return value*7;
222  }
223 ①↑    public double toWeek() {
224      return value;
225  }
226 ①↑    public double toMonth() {
227      return value/4;
228  }
229 ①↑    public double toYear() {
230      return value/52;
231  }
232 ①↑    public double toDecade() {
233      return value/520;
234  }
235 ①↑    public double toCentury() {
236      return value/5204;
237  }

239 ①↑ class Month extends Time {
240      public Month(double value) {
241          super(value);
242      }
243
244 ①↑    public double toSecond() {
245      return value*2592000;
246  }
247
248 ①↑    public double toMin() {
249      return value*43200;
250  }
251
252 ①↑    public double toHour() {
253      return value*720;
254  }
255 ①↑    public double toDay() {
256      return value*30;
257  }
258 ①↑    public double toWeek() {
259      return value*4;
260  }
261 ①↑    public double toMonth() {
262      return value;
263  }
```

```
264 ↑    public double toYear() {  Main.java × Time.java ×
265         return value/12;          return value*8760;
266     }                           }
267 ↑    public double toDecade() { 1 usage
268         return value/120;        public double toDay() {
269     }                           return value*365;
270 ↑    public double toCentury() { 1 usage
271         return value/1200;       public double toWeek() {
272     }                           return value*52;
273 }                           1 usage
274 class Year extends Time { 1 usage
275     public Year(double value) { 298 ↑    public double toMonth() {
276         super(value);           return value*12;
277     }                           299 ↑    public double toYear() {
278     }                           return value;
279 ↑    public double toSecond() { 300
280         return value*31536000; 301
281     }                           302 ↑    public double toDecade() {
282     }                           return value/10;
283 ↑    public double toMin() { 303
284         return value*525600;   304
285     }                           305 ↑    public double toCentury() {
286     }                           return value/100;
287 ↑    public double toHour() { 306
288         return value*86400;    307
289     }                           308
290 }                           309 class Decade extends Time {
291     public Decade(double value) { 310
292         super(value);
293     }                           311 }
```

```
Main.java x Time.java x Main.java x Time.java x
314 ①T public double toSecond() {
315     return value*315360000;
316 }
317
318 ①↑ 1 usage
319     public double toMin() {
320         return (value*5259492);
321     }
322 ①↑ 1 usage
323     public double toHour() {
324         return (value*87600);
325 ①↑ 1 usage
326     public double toDay() {
327         return value*3650;
328 ①↑ 1 usage
329     public double toWeek() {
330         return value* 521;
331 ①↑ 1 usage
332     public double toMonth() {
333         return value*120;
334 ①↑ 1 usage
335     public double toYear() {
336         return value*10;
337 ①↑ 1 usage
338     public double toDecade() {
339         return value;
340 ①↑ 1 usage
341     public double toCentury() {
342         return value/10;
343     }
344 class Century extends Time {
345     public Century(double value) {
346         super(value);
347     }
348
349 ①↑ 1 usage
350     public double toSecond() {
351         return value*315360000;
352     }
353 ①↑ 1 usage
354     public double toMin() {
355         return value*52596000;
356     }
357 ①↑ 1 usage
358     public double toHour() {
359         return value*87600;
360 ①↑ 1 usage
361     public double toDay() {
362         return value*365000;
363 ①↑ 1 usage
364     public double toWeek() {
365         return value*52103;
366 ①↑ 1 usage
367     public double toMonth() {
368         return value*12000;
369 ①↑ 1 usage
370     public double toYear() {
371         return value*100;
372 ①↑ 1 usage
373     public double toDecade() {
374         return value*10;
375 ①↑ 1 usage
376     public double toCentury() {
377         return value;
378 }
```

```
Main.java × Time.java ×  
2           82 usages  
3           protected double value;  
4           9 usages  
5           public Time (double value) {  
6           |   this.value = value;  
7           |}  
8           1 usage 9 implementations  
9           public abstract double toSecond();  
10          1 usage 9 implementations  
11          public abstract double toMin();  
12          1 usage 9 implementations  
13          public abstract double toHour();  
14          1 usage 9 implementations  
15          public abstract double toDay();  
16          1 usage 9 implementations  
17          public abstract double toWeek();  
18          1 usage 9 implementations  
19          public abstract double toMonth();  
20          1 usage 9 implementations  
21          public abstract double toYear();  
22          1 usage 9 implementations  
23          public abstract double toDecade();  
24          1 usage 9 implementations  
25          public abstract double toCentury();  
26      }
```

Currency Class

```
// Main class
import java.util.Scanner; // Import the Scanner class
no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Scanner Input = new Scanner(System.in); // Create instances of currency converters

        USDtoMYRCurrencyConverter usdtonmyrConverter = new USDtoMYRCurrencyConverter( rate: 4.47629 );
        USDtoGbpCurrencyConverter usdtogdpConverter = new USDtoGbpCurrencyConverter( rate: 0.843810 );
        USDtoWonCurrencyConverter usdtownonConverter = new USDtoWonCurrencyConverter( rate: 1315.0556 );
        USDtoYenCurrencyConverter usdtyenConverter = new USDtoYenCurrencyConverter( rate: 136.76646 );
        USDtoEuroCurrencyConverter usdteuroConverter = new USDtoEuroCurrencyConverter( rate: 0.94560212 );
```

```
14 // convert from USD to malaysian ringgit
15
16     Integer choice;
17     System.out.println("Please enter which converter (1 for USD, 2 for MYR, 3 for Euro,...): ");
18     choice = Input.nextInt();
19
20     if(choice == 1)
21     {
22         double usdAmount;
23         System.out.println("Please enter the amount: ");
24         usdAmount = Input.nextDouble();
25
26         Integer choice2;
27         System.out.println("Please enter which converter: ");
28         System.out.println("1 for GDB \n2 for MYR \n3 for Korean Won \n4 for Japanese Yen" +
29                           "\n5 for Euro");
30         choice2 = Input.nextInt();
31
32         if(choice2 == 2){
33             double MYRAmount = usdtonmyrConverter.convert(usdAmount);
34             System.out.println("$" + usdAmount + " is equivalent to RM" + MYRAmount);
35         } else if(choice2 == 1){
36             double GbpAmount = usdtogdpConverter.convert(usdAmount);
37             System.out.println("$" + usdAmount + " is equivalent to " + GbpAmount + " GBP");
38         } else if (choice2 == 3) {
39             double WonAmount = usdtownonConverter.convert(usdAmount);
40             System.out.println("$" + usdAmount + " is equivalent to " + WonAmount + " Korean won");
41         } else if(choice2 == 4){
42             double YenAmount = usdtyenConverter.convert(usdAmount);
43             System.out.println("$" + usdAmount + " is equivalent to " + YenAmount + " Japanese Yen");
44         } else if(choice2 == 5){
45             double EuroAmount = usdteuroConverter.convert(usdAmount);
46             System.out.println("$" + usdAmount + " is equivalent to " + EuroAmount + " Euros");
47         }
48     }
49 }
```

```
1 class CurrencyConverter {
2     protected double rate;
3
4     public CurrencyConverter(double rate) {
5         this.rate = rate;
6     }
7
8     public double convert(double amount) {
9         return amount * rate;
10    }
11 }
```

A screenshot of a Java code editor showing a single file named `USDTotEuroCurrencyConverter.java`. The code defines a class `USDTotEuroCurrencyConverter` that extends `CurrencyConverter`. It has a constructor that takes a double parameter and calls `super(rate)`. The class also contains a method `convertToEuro` that returns the amount converted.

```
1  class USDTotEuroCurrencyConverter extends CurrencyConverter{  
2      2 usages  
3      public USDTotEuroCurrencyConverter(double rate) {super(rate);}  
4  
5      no usages  
6      public double convertToEuro(double amount) {return convert(amount);}  
7  }  
8  
9  
10 |
```

Mass Class

The screenshot shows a Java IDE interface with the following details:

- Project:** Unicon
- File:** Main.java
- Content:** The code defines a Main class with a main method. The main method imports java.util.* and uses Scanner to read input from the user. It prints a menu of units (Milligram, Gram, Kilogram, Pound, Stones) and prompts the user to choose a unit to convert from. It then prompts the user to insert a value and reads it using nextInt(). Finally, it prints a menu of units to convert to and reads the chosen unit using nextLine().

```
import java.util.*;
no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        String unitFrom, unitTo;
        double value;

        System.out.println("[1]Milligram\n[2]Gram\n[3]Kilogram\n[4]Pound\n[5]Stones");
        System.out.print("[1/2/3/4/5] Choose unit to convert from: ");
        unitFrom = input.nextLine();

        System.out.println("Insert value: ");
        value = input.nextInt();

        System.out.println("[1/2/3/4/5] \nChoose unit to convert to: ");
        unitTo = input.nextLine();
    }
}
```

The screenshot shows a Java code editor interface with the following details:

- Project Bar:** Shows files Main.java, Gram.java (selected), Milligram.java, and Pounds.java.
- Gram.java Content:**

```
public class Gram {
    public double gToMg(float num){
        num = num * 1000;
        return num;
    }
    public double gToKg(float num){
        num = num / 1000;
        return num;
    }
    public double gToLbs(double num){
        num = num / 453.6;
        return num;
    }
    public double gToSt(float num){
        num = num / 6350;
        return num;
    }
}
```
- Left Sidebar:** Includes "Project" and "Bookmarks".
- Top Bar:** Includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, and Window menus.

The screenshot shows a Java code editor interface with the following details:

- Project:** UnIcon
- File:** Milligram.java
- Code Content:**

```
1  public class Milligram {  
2      public double mgToG(double num){  
3          num = num / 1000;  
4          return num;  
5      }  
6  
7      public double mgToKg(float num){  
8          num = num / 1000 / 1000;  
9          return num;  
10     }  
11  
12     public double mgToLbs(double num){  
13         num = num / 453600;  
14         return num;  
15     }  
16  
17     public double mgToSt(float num){  
18         num = num / 6350000;  
19         return num;  
20     }  
21  
22 }  
23
```

- Toolbars:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, Help
- Status Bar:** no usages

The screenshot shows a Java code editor with the following interface:

- Menu Bar:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools.
- Project Tree:** Shows a single project named "Unicon".
- Tab Bar:** Main.java, Gram.java, Milligram.java, and Pounds (which is the active tab).
- Code Editor:** Displays the content of the Pounds.java file.

```
public class Pounds {
    public double lbsToMg(float num){
        num = num * 453600;
        return num;
    }

    public double lbsToG(double num){
        num = num * 453.6;
        return num;
    }

    public double lbsToKg(double num){
        num = num / 453.6;
        return num;
    }

    public double lbsToSt(float num){
        num = num / 14;
        return num;
    }
}
```

The screenshot shows a Java code editor interface with the following details:

- Project:** Unicon
- File:** src/Stones.java
- Code:**

```
1 public class Stones {  
2     |  
3     no usages  
4     public double stToMg(float num){  
5         num = num * 6350000;  
6         return num;  
7     }  
8     no usages  
9     public double stToG(float num){  
10        num = num * 6350;  
11        return num;  
12    }  
13    no usages  
14    public double stToKg(double num){  
15        num = num * 6.35;  
16        return num;  
17    }  
18    no usages  
19    public double stToLbs(float num){  
20        num = num * 14;  
21        return num;  
22    }  
23}
```

- Toolbars:** File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools
- Bottom Bar:** Bookmarks

The screenshot shows a Java code editor interface with the following details:

- Project:** Unicon
- File:** src > Stones
- Code Content:**

```
1  no usages
2  public class Stones {
3      no usages
4          public double stToMg(float num){
5              num = num * 6350000;
6              return num;
7          }
8
9          no usages
10         public double stToG(float num){
11             num = num * 6350;
12             return num;
13         }
14
15         no usages
16         public double stToKg(double num){
17             num = num * 6.35;
18             return num;
19         }
20
21         no usages
22         public double stToLbs(float num){
23             num = num * 14;
24             return num;
25         }
26     }
```

- Bookmarks:** A vertical bar on the left side of the code editor.

The screenshot shows a Java code editor interface with the following details:

- File Menu:** File Edit View Navigate Code Refactor Build Run Tools
- Project Path:** Unicon > src > Kilogram
- Open Files:** Main.java, Gram.java, Milligram.java, Pound.java
- Code Editor Content:**

```
no usages
public class Kilogram {
    no usages
    public double kgToMg(float num){
        num = num * 1000 * 1000;
        return num;
    }

    no usages
    public double kgToG(float num){
        num = num * 1000;
        return num;
    }

    no usages
    public double kgToLbs(double num){
        num = num * 2.205;
        return num;
    }

    no usages
    public double kgToSt(double num){
        num = num / 6.35;
        return num;
    }
}
```
- Left Sidebar:** Project, Bookmarks

Volume Class

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner input = new Scanner(System.in);
6
7         System.out.print("1. (M) Millilitre \n2. (L) Litre \n3. (O) Ounce");
8         System.out.print("Select unit(M/L/O): ");
9         String unit = input.nextLine().toUpperCase();
10        String unit2;
11
12        double measurement;
13
14        switch (unit){
15            case "M":
16                Millilitre millilitre = new Millilitre();
17
18                System.out.print("Enter millilitre value: ");
19                measurement = input.nextDouble();
20
21                System.out.print("Select unit to convert to(L/O): ");
22                unit2 = input.nextLine().toUpperCase();
23
24                switch (unit2){
25                    case "L":
26                        System.out.println("\n" + measurement + " millilitre in litre: " + String.format("%.4f", millilitre.millilitreToLitres(measurement)));
27                        break;
28                    case "O":
29                        System.out.println("\n" + measurement + " millilitre in ounce: " + String.format("%.4f", millilitre.millilitreToOunces(measurement)));
30                        break;
31                }
32            break;
33
34            case "L":
35
36                Litre litre = new Litre();
37
38                System.out.print("Enter litre value: ");
39                measurement = input.nextDouble();
40
41                System.out.print("Select unit to convert to(M/O): ");
42                unit2 = input.nextLine().toUpperCase();
43
44                switch (unit2){
45                    case "M":
46                        System.out.println("\n" + measurement + " litre in millilitre: " + String.format("%.4f", litre.litreToMillilitre(measurement)));
47                        break;
48                    case "O":
49                        System.out.println("\n" + measurement + " litre in ounce: " + String.format("%.4f", litre.litreToOunces(measurement)));
50                        break;
51                }
52            break;
53
54            case "O":
55                Ounce ounce = new Ounce();
56
57                System.out.print("Enter ounce value: ");
58                measurement = input.nextDouble();
59
60                System.out.print("Select unit to convert to(M/L): ");
61                unit2 = input.nextLine().toUpperCase();
62
63                switch (unit2){
64                    case "M":
65                        System.out.println("\n" + measurement + " ounce in millilitre: " + String.format("%.4f", ounce.ounceToMillilitre(measurement)));
66                        break;
67                    case "L":
68                        System.out.println("\n" + measurement + " ounce in litre: " + String.format("%.4f", ounce.ounceToLitres(measurement)));
69                        break;
70                }
71            break;
72        }
73    }
74}
```

```
1 Main.java  Volume.java
2
3 public class Volume {
4     protected double measurement;
5
6     public void setMeasurement(double measurement){
7         this.measurement = measurement;
8     }
9
10    class Litre extends Volume{
11        private String unit = "Litres";
12
13        public String getUnit() { return this.unit; }
14
15        public double litreToMilliliter(double measurement){
16            double answer;
17            answer = measurement * 1000;
18            return answer;
19        }
20
21        public double litreToDunce(double measurement){
22            double answer;
23            answer = measurement * 33.81;
24            return answer;
25        }
26    }
27
28
29    class Millilitre extends Volume{
30        private String unit = "Millilitre";
31
32        public String getUnit() { return this.unit; }
33
34        public double millilitreToLitre(double measurement){
35            double answer;
36            answer = measurement / 1000;
37            return answer;
38        }
39
40        public double millilitreToDunce(double measurement){
41            double answer;
42            answer = measurement / 29.57;
43            return answer;
44        }
45    }
46
47
48    class Dunce extends Volume{
49        private String unit = "Dunce";
50
51        public String getUnit() { return this.unit; }
52
53        public double ounceToLitre(double measurement){
54            double answer;
55            answer = measurement / 33.814;
56            return answer;
57        }
58
59        public double ounceToMillilitre(double measurement){
60            double answer;
61            answer = measurement * 29.574;
62            return answer;
63        }
64    }
65
66
67
68 }
```

Area Class

```
1 import java.util.*;  
2  
3 public class Main {  
4  
5     public static void main(String[] args) {  
6         double squareMeters = 10;  
7         double squareFeet = 100;  
8  
9         AreaConverter squareMeterToSquareFootConverter = new SquareMeterToSquareFootConverter();  
10        AreaConverter squareFootToSquareMeterConverter = new SquareFootToSquareMeterConverter();  
11  
12        double convertedSquareFeet = squareMeterToSquareFootConverter.convert(squareMeters);  
13        double convertedSquareMeters = squareFootToSquareMeterConverter.convert(squareFeet);  
14  
15        System.out.println(squareMeters + " square meters is equal to " + convertedSquareFeet + " square feet");  
16        System.out.println(squareFeet + " square feet is equal to " + convertedSquareMeters + " square meters");  
17    }  
18}
```

```
1 10 usages 8 inheritors  
2 abstract class AreaConverter {  
3     8 implementations  
4     public abstract double convert(double area);  
5 }
```

```
1 usage  
2 public class SquareMeterToSquareFootConverter extends AreaConverter {  
3     1 usage  
4     private static final double CONVERSION_FACTOR = 10.76391;  
5  
6     @Override  
7     public double convert(double area) { return area * CONVERSION_FACTOR; }  
8 }
```

```
1 usage  
2 public class SquareFootToSquareMeterConverter extends AreaConverter {  
3     1 usage  
4     private static final double CONVERSION_FACTOR = 0.09290304;  
5  
6     @Override  
7     public double convert(double area) { return area * CONVERSION_FACTOR; }  
8 }
```

```
C SquareCentimeterToSquareMilimeterConverter.java × C SquareCentimeterToSquareMeter.java × C SquareMeterToSquareCenti
no usages
public class SquareCentimeterToSquareMilimeterConverter extends AreaConverter {
    1 usage
    private static final double CONVERSION_FACTOR = 100;

    @Override
    public double convert(double area) { return area * CONVERSION_FACTOR; }
}
```

```
C SquareCentimeterToSquareMilimeterConverter.java × C SquareCentimeterToSquareMeter.java × C Square
no usages
public class SquareCentimeterToSquareMeter extends AreaConverter {
    1 usage
    private static final double CONVERSION_FACTOR = 0.0001;

    @Override
    public double convert(double area) {
        return area * CONVERSION_FACTOR;
    }
}
```

```
va × C SquareMeterToSquareCentimeterConverter.java × C SquareMilimeterToSquareCentimeter.java × C SquareKilome
no usages
public class SquareMeterToSquareCentimeterConverter extends AreaConverter {
    1 usage
    private static final double CONVERSION_FACTOR = 10000;

    @Override
    public double convert(double area) {
        return area * CONVERSION_FACTOR;
    }
}
```

```
× C SquareMeterToSquareCentimeterConverter.java × C SquareMilimeterToSquareCentimeter.java × C SquareKilometerToSquareInchesConverter.java ×
no usages
public class SquareMilimeterToSquareCentimeter extends AreaConverter {

    1 usage
    private static final double CONVERSION_FACTOR = 0.01;

    @Override
    public double convert(double area) { return area * CONVERSION_FACTOR; }
}
```

```
a × C SquareMeterToSquareCentimeterConverter.java × C SquareMilimeterToSquareCentimeter.java × C SquareKilometerToSquareInchesConverter.java ×
no usages
public class SquareKilometerToSquareInchesConverter extends AreaConverter {
    1 usage
    private static final double CONVERSION_FACTOR = 1550003100;

    @Override
    public double convert(double area) { return area * CONVERSION_FACTOR; }
}
```

```
a x  C SquareMeterToSquareCentimeterConverter.java x  C SquareMillimeterToSquareCentimeter.java x  C SquareKilometerToSquareInchesConverter.java x  C SquareInchesToSquareKilometer.java
no usages
public class SquareInchesToSquareKilometer extends AreaConverter {
    1 usage
    private static final double CONVERSION_FACTOR = 0.0000000064516;

    @Override
    public double convert(double area) { return area * CONVERSION_FACTOR; }
}
```

Memory Size Class

```
>Main.java < Bytes.java <
1  public class Bytes {
2
3      static long size;
4
5      static long kilo = 1024;
6      static long mega = kilo * kilo;
7      static long giga = mega * kilo;
8      static long tera = giga * kilo;
9
10
11     public void bitsToByte(long size){
12         this.size = size;
13
14         double byteSize = (double)size / 8;
15
16         System.out.println(byteSize + " byte");
17
18         byteToKb(byteSize);
19         byteToMb(byteSize);
20         byteToGb(byteSize);
21         byteToTb(byteSize);
22     }
23
24     public void byteToKb(double size){
25         double kb = size / kilo;
26
27         System.out.println(kb + " KB");
28     }
29
30     public void byteToMb(double size){
31         double mb = size / mega;
32
33         System.out.println(mb + " MB");
34     }
35
36
37     public void byteToGb(double size){
38         double gb = size / giga;
39
40         System.out.println(gb + " GB");
41     }
42
43     public void byteToTb(double size) {
44         double tb = size / tera;
45
46         System.out.println(tb + " TB");
47     }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
778
779
779
780
781
782
783
784
785
786
787
788
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
928
928
929
929
930
931
932
933
934
935
936
937
938
938
939
939
940
941
942
943
944
945
946
947
947
948
949
949
950
951
952
953
954
955
956
957
958
958
959
959
960
961
962
963
964
965
966
967
967
968
968
969
969
970
971
972
973
974
975
976
977
977
978
978
979
979
980
981
982
983
984
985
986
987
987
988
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1705
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1805
1805
1806
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1856
1857
1857
1858
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1866
1867
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1876
1877
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1886
1887
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1895
1896
1896
1897
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1905
1906
1906
1907
1907
1908
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1916
1917
1917
1918
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1926
1927
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1936
1937
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1946
1947
1947
1948
1948
1949
1949
1950
1951
1952
1953
19
```

Pressure Class

```
1 import java.text.DecimalFormat;
2 import java.text.NumberFormat;
3 import java.util.*;
4 
5 public class PressureConverter {
6     public static void main(String[] args) {
7         Scanner input = new Scanner(System.in);
8         NumberFormat formatter = new DecimalFormat(pattern:"#0.00");
9 
10        System.out.print(s: "Enter Pressure Value: ");
11        double pressureValue = input.nextDouble();
12 
13        System.out.print(s: "Enter Pressure Unit (A/P): ");
14        String initialUnit = input.next();
15 
16        System.out.print(s: "Enter unit to convert to (A/P): ");
17        String finalUnit = input.next();
18        input.close();
19 
20        Pressure initialPressure;
21        if (initialUnit.equalsIgnoreCase(anotherString: "A")) {
22            initialPressure = new Atmosphere(pressureValue);
23        } else if (initialUnit.equalsIgnoreCase(anotherString: "P")) {
24            initialPressure = new Pascal(pressureValue);
25        } else {
26            System.out.print(s: "Invalid Pressure Unit.");
27            return;
28        }
29 
30        double convertedPressure;
31        if (finalUnit.equalsIgnoreCase(anotherString: "A")) {
32            convertedPressure = initialPressure.toAtmosphere();
33            System.out.println("Pressure in Atmosphere(atm): " + formatter.format(convertedPressure));
34        } else if (finalUnit.equalsIgnoreCase(anotherString: "P")) {
35            convertedPressure = initialPressure.toPascal();
36            System.out.println("Pressure in Pascal(Pa): " + formatter.format(convertedPressure));
37        } else {
38            System.out.print(s: "Invalid Pressure Unit.");
39            return;
40        }
41    }
42 }
43 }
```

```
45  class Atmospshere extends Pressure {  
46      no usages  
47      public Atmospshere(double value) {  
48          super(value);  
49      }  
50      1 usage  
51      public double toAtmosphere() {  
52          return value;  
53      }  
54      1 usage  
55      public double toPascal() {  
56          return (value * 101325);  
57      }  
58  class Pascal extends Pressure {  
59      1 usage  
60      public Pascal(double value) {  
61          super(value);  
62      }  
63      1 usage  
64      public double toAtmosphere() {  
65          return (value / 101325);  
66      }  
67      1 usage  
68      public double toPascal() {  
69          return (value);  
70      }  
    }
```

PressureConverter.java × Pressure.java ×

Project JDK is not defined

```
1  ⚒ 3 usages 2 inheritors
2    ⚒ abstract class Pressure {
3      5 usages
4        protected double value;
5
6        2 usages 2 related problems
7          public Pressure(double value) {
8            this.value = value;
9          }
10         1 usage 2 implementations
11           ⚒ public abstract double toAtmosphere();
12
13         1 usage 2 implementations
14           public abstract double toPascal();
15         }
```