```
/*
    Some improvements needed for toString methods -0.3
    Fully explore the super.toString
    Avoid subclasses access to super class info as much as you can

    abstract volume class -0.1

    Overall design OK

    Display on screen OK

    Use do-while loop if the number of objects subject to
    certain conditions

    In the display list   -0.3
       step 1:  display the shape (no testing is required)
       step 2:  display the area  (no testing is required)
       step 3: if (ThreeD)
          -> display the volume (no testing of specific 3 D)
      - step 4: detail shape (use instanceof)
      - Conclusion: Let Java enjoy the polymorphisms

    5.3 / 6
    Marks allocated subject to demo and you pass the plagiarism test
*/



// Full Name - Hay Munn Hnin Wai
// Assignment-2
// Tutorial - T04
// Dear Sir, This is my own work & Kindly check the below code with some comments.

import java.util.ArrayList;
import java.util.Random;
import java.lang.Math.*;

enum ShapeColor
{
    Blue,
    Yellow,
    Red,
    Green,
    White

};
 // Interface -> Shape
interface Shape
{
    public double area();

}
// Abstract Class -> TwoD
abstract class TwoD implements Shape
{

        protected ShapeColor sc;
        protected int a;
        protected int b;
        protected int c;
        protected int d;

     // Constructor (s)
    public TwoD()
    {
        sc = ShapeColor.Blue;
        a = 1;
        b = 1;
        c = 1;
        d = 1;
```

```
71
72          }
73          public TwoD (ShapeColor sc, int a)
74          {
75              this.sc = sc;
76              this.a = a;
77
78          }
79          public TwoD (ShapeColor sc, int a , int b)
80          {
81              this.sc = sc;
82              this.a = a;
83              this.b = b;
84
85          }
86          public TwoD (ShapeColor sc, int a , int b, int c)
87          {
88              this.sc = sc;
89              this.a = a;
90              this.b = b;
91              this.c = c;
92
93          }
94          public TwoD (ShapeColor sc, int a , int b, int c, int d)
95          {
96              this.sc = sc;
97              this.a = a;
98              this.b = b;
99              this.c = c;
100             this.d = d;
101
102         }
103         // Copy Constructor
104         public TwoD ( TwoD td)
105         {
106             this(td.sc,td.a,td.b,td.c,td.d );
107
108         }
109         // Accessor Methods
110         public int getA()
111         {
112             return a;
113
114         }
115         public int getB()
116         {
117             return b;
118
119         }
120         public int getC()
121         {
122             return c;
123
124         }
125         public int getD()
126         {
127             return d;
128
129         }
130         public ShapeColor getShapeColor()
131         {
132             return sc;
133
134
135         }
136         public void set (ShapeColor sc, int a, int b, int c, int d)
137         {
138             this.sc = sc;
139             this.a = a;
140             this.b = b;
141             this.c = c;
```

```
142             this.d = d;
143 
144         }
145         @Override
146         public String toString()
147         {
148             return String.format("");
149 
150 
151         }
152     }
153     // Sub-Class of TwoD
154     class Circle extends TwoD
155     {
156         public Circle()
157         {
158             sc = ShapeColor.Blue;
159 
160         }
161         public Circle(ShapeColor sc, int radius)
162         {
163             super(sc,radius);
164 
165         }
166             // Copy Constructor
167         public Circle (Circle c)
168         {
169             this(c.sc,c.a);
170         }
171 
172         private double computeArea ()
173         {
174             return Math.PI * a * a;
175         }
176         @Override
177         public double area ()
178         {
179             return this.computeArea();
180         }
181         public int getRadius()
182         {
183             return a;
184 
185         }
186         public void set(ShapeColor sc, int radius)
187         {
188             this.sc = sc;
189             this.a = a;
190 
191         }
192 
193         @Override
194         public String toString()
195         {
196             return String.format("%sCircle (2D (%s, %d))",
197                                  super.toString (), sc, a);
198 
199         }
200 
201     }
202      // Sub Class of TwoD
203     class Triangle extends TwoD
204     {
205         public Triangle()
206         {
207             sc = ShapeColor.Blue;
208             a = 1;
209             b = 1;
210             c = 1;
211 
212         }
```

```
213         public Triangle (ShapeColor sc, int a, int b, int c)
214         {
215             super (sc, a , b, c);
216         }
217         public Triangle ( Triangle t)
218         {
219             this(t.sc,t.a,t.b,t.c);
220 
221         }
222 
223         private double computeArea ()
224         {
225             double s = (a + b + c)/2.0;
226             return Math.sqrt(s *(s - a) *(s - b) *(s - c));
227         }
228         @Override
229         public double area()
230         {
231 
232             return this.computeArea();
233 
234         }
235         public int getA()
236         {
237             return a;
238         }
239         public int getB()
240         {
241             return b;
242 
243         }
244         public int getC()
245         {
246             return c;
247         }
248         public void set(ShapeColor sc, int a, int b, int c)
249         {
250             this.sc = sc;
251             this.a = a;
252             this.b = b;
253             this.c = c;
254 
255         }
256         @Override
257         public String toString()
258         {
259             return String.format("%sTriangle (2D (%s, %d,%d,%d))",
260                                  super.toString (), sc, a,b,c);
261         }
262 
263     }
264 
265     class Rectangle extends TwoD
266     {
267         public Rectangle()
268         {
269             sc = ShapeColor.Blue;
270             a = 1;
271             b = 1;
272 
273         }
274         public Rectangle (ShapeColor sc, int length, int width)
275         {
276             super(sc,length, width);
277 
278         }
279         public Rectangle ( Rectangle r)
280         {
281             this(r.sc, r.a, r.b);
282 
283         }
```

```
284
285         private double computeArea ()
286         {
287             return a * b;
288         }
289         @Override
290         public double area()
291         {
292             return this.computeArea();
293
294         }
295         public int getLength()
296         {
297             return a;
298
299         }
300         public int getWidth()
301         {
302             return b;
303
304         }
305         public void set ( ShapeColor sc, int length, int width)
306         {
307             this.sc = sc;
308             this.a = a;
309             this.b = b;
310
311         }
312         @Override
313         public String toString()
314         {
315             return String.format("%sRectangle (2D (%s, %d,%d))",
316                                  super.toString (), sc, a,b);
317
318         }
319
320     }
321     class Trapezoid extends TwoD
322     {
323         private int h;
324         public Trapezoid ()
325         {
326             sc = ShapeColor.Blue;
327             a = 1;
328             b = 1;
329             c = 1;
330             d = 1;
331             h = 1;
332
333         }
334         public Trapezoid ( ShapeColor sc, int a, int b, int c, int d, int h )
335         {
336             super(sc,a,b,c,d);
337             this.h = h;
338
339         }
340         public int getA()
341         {
342             return a;
343
344         }
345         public int getB()
346         {
347             return b;
348
349         }
350         public int getC()
351         {
352             return c;
353
354         }
```

```
355         public int getD()
356         {
357             return d;
358 
359         }
360         public int getHeight()
361         {
362             return h;
363 
364         }
365         public void setHeight(int h)
366         {
367             this.h = h;
368         }
369 
370         private double computeArea ()
371         {
372             return ((a+b)*h)/2 ;
373         }
374         @Override
375         public double area()
376         {
377             return this.computeArea();
378         }
379         public void set(ShapeColor sc, int a, int b, int c,int d)
380         {
381             this.sc = sc;
382             this.a = a;
383             this.b = b;
384             this.c = c;
385             this.d = d;
386 
387         }
388         @Override
389         public String toString()
390         {
391             return String.format("%sTrapezoid (2D (%s, %d,%d,%d,%d),%d)",
392                                  super.toString (), sc, a,b,c,d,h);
393 
394         }
395 
396     }
397     // Interface ---> Resizable for 3D
398     interface Resizable
399     {
400         public void resize();
401 
402     }
403       //*********ThreeD*************//
404     abstract class ThreeD implements Shape , Resizable
405     {
406         protected ShapeColor sc;
407         protected double a;
408 
409         public ThreeD()
410         {
411             sc = ShapeColor.Blue;
412             a = 1.0;
413 
414         }
415         public ThreeD (ShapeColor sc, double a )
416         {
417             this.sc = sc;
418             this.a = a;
419 
420         }
421         public double getA()
422         {
423             return a;
424 
425         }
```

```
426         public void set(ShapeColor sc, double a)
427         {
428             this.sc = sc;
429             this.a = a;
430 
431         }
432          // Reduce by 10% from Original double value (a)
433         public void resize()
434         {
435             a = a * 0.9;
436 
437         }
438          //Volume Method
439         protected double volume()
440         {
441             return 0.0;
442         }
443 
444         @Override
445         public String toString()
446         {
447             return String.format("");
448         }
449     }
450     // Class Sphere
451     class Sphere extends ThreeD
452     {
453         public Sphere()
454         {
455             sc = ShapeColor.Blue;
456             a = 1.0;
457 
458         }
459         public Sphere( ShapeColor sc, double a)
460         {
461             this.sc = sc;
462             this.a = a;
463 
464         }
465         public Sphere (Sphere s)
466         {
467             this(s.sc,s.a);
468 
469         }
470 
471         private double computeArea ()
472         {
473             return 4*Math.PI*a*a;
474         }
475         @Override
476         public double area()
477         {
478             return this.computeArea();
479         }
480 
481         private double computeVolume ()
482         {
483             return 4/3 * Math.PI * a*a*a;
484         }
485         @Override
486         protected double volume()
487         {
488             return this.computeVolume();
489 
490         }
491         public double getA()
492         {
493             return a;
494 
495         }
496         public void set(ShapeColor sc, double a)
```

```
497         {
498             this.sc = sc;
499             this.a = a;
500         }
501         public String toString()
502         {
503             return String.format("%sSphere( 3D
      (%s,%.3f))",super.toString(),sc,a);
504 
505         }
506 
507     }
508      // Class --> Cube
509     class Cube extends ThreeD
510     {
511         public Cube ()
512         {
513             sc = ShapeColor.Blue;
514             a = 1.0;
515 
516         }
517         public Cube (ShapeColor sc, double a)
518         {
519             this.sc = sc;
520             this.a = a ;
521 
522         }
523         public Cube ( Cube c)
524         {
525             this(c.sc, c.a);
526 
527         }
528 
529         private double computeArea ()
530         {
531             return 6*(a*a);
532         }
533         @Override
534         public double area()
535         {
536             return this.computeArea();
537 
538         }
539 
540         private double computeVolume ()
541         {
542             return a*a*a;
543         }
544         @Override
545         protected double volume()
546         {
547             return this.computeVolume();
548 
549         }
550         public double getA()
551         {
552             return a;
553         }
554         public void set(ShapeColor sc, double a)
555         {
556             this.sc = sc;
557             this.a = a;
558 
559         }
560         public String toString()
561         {
562             return String.format("%sCube( 3D (%s,%.3f))",super.toString(),sc,a
563         }
564 
565     }
566     //Class Tetrahedron
```

```
567   class Tetrahedron extends ThreeD
568   {
569       public Tetrahedron()
570       {
571           sc = ShapeColor.Blue;
572           a = 1.0;
573
574       }
575       public Tetrahedron(ShapeColor sc, double a)
576       {
577           this.sc = sc;
578           this.a = a;
579
580       }
581       public Tetrahedron(Tetrahedron t)
582       {
583           this(t.sc, t.a);
584
585       }
586
587       private double computeArea ()
588       {
589           return Math.sqrt(3)*(a*a);
590       }
591       @Override
592       public double area()
593       {
594           return this.computeArea();
595
596       }
597
598       private double computeVolume ()
599       {
600           return (a*a*a)/(6*Math.sqrt(2));
601       }
602       @Override
603       protected double volume()
604       {
605           return this.computeVolume();
606
607       }
608       public double getA()
609       {
610           return a;
611
612       }
613       public void set(ShapeColor sc, double a)
614       {
615           this.sc = sc;
616           this.a = a;
617
618       }
619       @Override
620       public String toString()
621       {
622           return String.format("%sTetrahedron( 3D
      (%s,%.3f))",super.toString(),sc,a);
623
624       }
625   }
626   class HayMunnHninWai_60_A2
627   {
628       private static int getInt()
629       {
630           int k = (int) (Math.random () * 5) + 1;
631           return k;
632
633       }
634       private static double getDouble()
635       {
636           double i = (int) (Math.random () * 5.0) + 1.5;
```

```
637             return i;
638         }
639         private static ShapeColor getColor()
640         {
641             Random random = new Random();
642 
643             ShapeColor getColor =
      ShapeColor.values()[random.nextInt(ShapeColor.values().length)];
644             return getColor;
645 
646         }
647         private static boolean isTriangle( int a, int b, int c)
648         {
649         if((a+b)>c && (a+c)>b && (b+c)>a)
650             return true;
651         else
652             return false;
653 
654         }
655         private static TwoD getTwoD()
656         {
657             int i = (int)((Math.random()* 4)+ 0.5);
658             int x, y, z;
659 
660             Circle c = new Circle (getColor(),getInt());
661             Rectangle rc = new Rectangle (getColor(),getInt(), getInt());
662 
663             Trapezoid tr = new Trapezoid(getColor(), getInt(), getInt(),
664                                          getInt(), getInt(), getInt());
665 
666             while (true)
667             {
668                 x = getInt();
669                 y = getInt();
670                 z = getInt();
671 
672                 boolean b = isTriangle (x,y,z);    // Check isTriangle or not
673                 if(b)
674                 {
675                     break;
676                 }
677             }
678             Triangle t = new Triangle (getColor(),x, y, z);
679 
680             switch(i)
681             {
682                 case 1: return c;
683                 case 2: return rc;
684                 case 3: return t;
685                 case 4: return tr;
686                 default: return c;
687             }
688 
689         }
690         private static ThreeD getThreeD()
691         {
692             int i1 = (int)((Math.random()* 4)+ 0.5);
693             double d;
694 
695             Sphere s = new Sphere (getColor(), getDouble());
696             Cube cb = new Cube (getColor(), getDouble());
697             Tetrahedron tt = new Tetrahedron (getColor(), getDouble());
698 
699             switch (i1)
700             {
701                 case 1: return s;
702                 case 2: return cb;
703                 case 3: return tt;
704                 default : return cb;
705             }
706         }
```

```
707       //display from alist
708       private static void displayList (ArrayList <Shape> alist)
709       {
710           int count = 1;
711           for (Shape s : alist)
712           {
713               if (count > 4)
714                   break;
715               System.out.printf("Shape %d: ",count++);
716
717
718               if (s instanceof Circle ||s instanceof Rectangle ||
719                   s instanceof Triangle ||s instanceof Trapezoid)
720               {
721                   System.out.println(s);
722                   System.out.printf ("Area = %.3f%n", s.area ());
723
724                   if (s instanceof Circle )
725               {
726                   System.out.printf("I am a %s shape%n", "Circle");
727               }
728               else if ( s instanceof Triangle )
729               {
730                   System.out.printf("I am a %s shape%n", "Triangle");
731
732               }
733               else if ( s instanceof Rectangle)
734               {
735                   System.out.printf("I am a %s shape%n", "Rectangle");
736               }
737
738               else if ( s instanceof Trapezoid)
739               {
740                   System.out.printf("I am a %s shape%n", "Trapezoid");
741               }
742
743               System.out.println ("---------------------------------");
744
745               }
746               else                  // For 3D Objects
747               {
748                   System.out.println(s);                                  // Print
      Elements
749                   System.out.printf ("Surface area = %.3f%n", s.area ()); //
      Print Area()
750               if (s instanceof Sphere )
751               {
752                   Sphere ss = (Sphere)(s);
753                   System.out.printf("Volume = %.3f%n", ss.volume());
754                   ss.resize();
755                   System.out.printf("Size reduced by 10%% :
      %s%n",ss.toString());
756                   System.out.printf("Updated surface area = %.3f%n",ss.area(
757                   System.out.printf("Updated volume =  %.3f%n",ss.volume());
758                   System.out.printf("I am a %s shape%n","Sphere");
759
760               }
761               else if (s instanceof Cube )
762               {
763                   Cube cc = (Cube)(s);
764                   System.out.printf("Volume = %.3f%n", cc.volume());
765                   cc.resize();
766                   System.out.printf("Size reduced by 10%% :
      %s%n",cc.toString());
767                   System.out.printf("Updated surface area = %.3f%n",cc.area(
768                   System.out.printf("Updated volume =  %.3f%n",cc.volume());
769                   System.out.printf("I am a %s shape%n","Cube");
770
771               }
772               else
773               {
```

```
774                     Tetrahedron tt = (Tetrahedron)(s);
775                     System.out.printf("Volume = %.3f%n", tt.volume());
776                     tt.resize();
777                     System.out.printf("Size reduced by 10%%:
      %s%n",tt.toString());
778                     System.out.printf("Updated surface area = %.3f%n",tt.area(
779                     System.out.printf("Updated volume =  %.3f%n",tt.volume());
780                     System.out.printf("I am a %s shape%n","Tetrahedron");
781                 }
782 
783                 System.out.println ("---------------------------------");
784 
785                 }
786             }
787         }
788         public static void main ( String [] args)
789         {
790             ArrayList <Shape> alist = new ArrayList <Shape> ();
791 
792             int k = 1;
793             int i = 1;
794             while ( k>0)
795             {
796                 k = (int)(Math.random()*2+1);
797 
798                 switch (k)
799                 {
800                     case 1: alist.add (getTwoD());
801                     case 2: alist.add (getThreeD());
802                     default: break;
803                 }
804                 if (i > 4)
805                     k = 0;
806                 i++;
807             }
808 
809             displayList(alist);
810         }
811     }
812 
813 
814 
815 
816 
817 
818 
819 
820 
821 
822 
823 
824 
825 
826 
827 
828 
829 
830 
831 
832 
833 
834 
835 
836 
837 
```