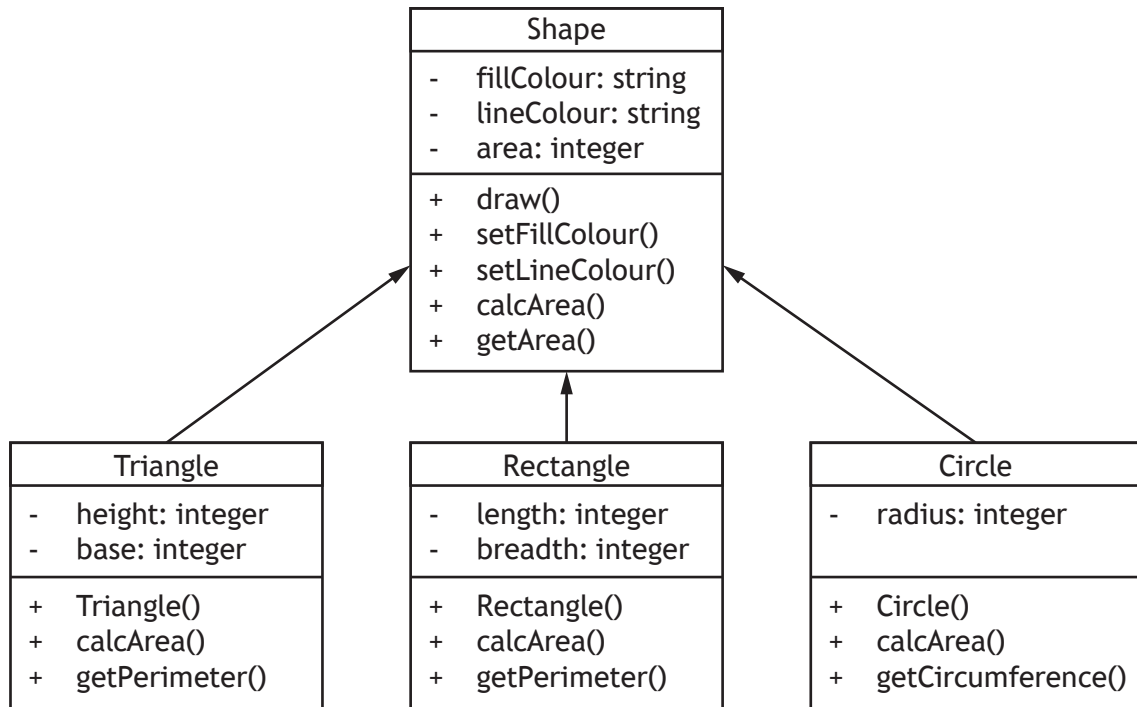


5. Object-oriented code is used to create a drawing app for children.

The simplified UML class diagram below represents the classes used in the app.



Some of the code used to implement the drawing app is provided below.

```
CLASS Shape { STRING fillColour, STRING lineColour, INTEGER area}
```

METHODS

```
CONSTRUCTOR ( STRING fill, STRING line )
    DECLARE THIS.fillColour TO fill
    DECLARE THIS.lineColour TO line
    DECLARE THIS.area TO 0
END CONSTRUCTOR

...

PROCEDURE setFillColour ( STRING colour )
    SET THIS.fillColour TO colour
END PROCEDURE

PROCEDURE calcArea()
    SEND "The area is the space enclosed within the boundary of
    the shape" TO DISPLAY
END PROCEDURE

FUNCTION getArea() RETURNS INTEGER
    RETURN THIS.area
END FUNCTION

END CLASS
```

## 5. (continued)

```
CLASS Triangle INHERITS Shape WITH { INTEGER height, INTEGER  
base }
```

```
METHODS
```

```
...
```

```
  OVERRIDE PROCEDURE calcArea()  
    SET THIS.area TO THIS.height * THIS.base / 2  
  END PROCEDURE
```

```
...
```

```
END CLASS
```

```
CLASS Rectangle INHERITS Shape WITH { INTEGER length, INTEGER  
breadth }
```

```
METHODS
```

```
  CONSTRUCTOR Rectangle ( STRING fill, STRING line, INTEGER  
length, INTEGER breadth )
```

```
    DECLARE THIS.fillColour TO fill  
    DECLARE THIS.lineColour TO line  
    DECLARE THIS.length TO length  
    DECLARE THIS.breadth TO breadth  
    DECLARE THIS.area TO 0
```

```
  END CONSTRUCTOR
```

```
  OVERRIDE PROCEDURE calcArea()  
    SET THIS.area TO THIS.length * THIS.breadth  
  END PROCEDURE
```

```
...
```

```
END CLASS
```

```
CLASS Circle INHERITS Shape WITH { INTEGER radius }
```

```
METHODS
```

```
  CONSTRUCTOR Circle ( STRING fill, STRING line, INTEGER  
radius )
```

```
    DECLARE THIS.fillColour TO fill  
    DECLARE THIS.lineColour TO line  
    DECLARE THIS.radius TO radius  
    DECLARE THIS.area TO 0
```

```
  END CONSTRUCTOR
```

```
  OVERRIDE PROCEDURE calcArea()  
    SET THIS.area TO 3.14 * THIS.radius^2  
  END PROCEDURE
```

```
...
```

```
END CLASS
```

**5. (continued)**

- (a) With reference to the UML class diagram and code, explain the use made of overriding.

**2**

- (b) The following code appears within the main program that controls the app.

```
DECLARE r1 AS Rectangle ("white", "black", 20, 16)
```

- (i) Using appropriate object-oriented terminology, explain the effect of this statement.

**2**

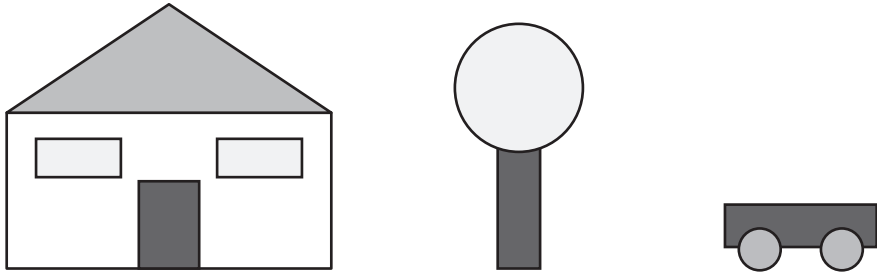
- (ii) State the output that would be produced by the following code:

**2**

```
SEND "The area is " & r1.getArea() TO DISPLAY  
r1.calcArea()  
SEND "The area is " & r1.getArea() TO DISPLAY
```

5. (continued)

- (c) A young child uses the app to create a simple image consisting of a house, a tree and a wagon. The final image is shown below.



Within the app, an array of `Shape` objects called `list` is used to keep track of the individual shapes used in the image:

<i>Index</i>	<i>Object referenced</i>	<i>Where used in the image</i>
0	r1	House building
1	r2	Window left
2	r3	Window right
3	r4	Door
4	t1	Roof
5	r5	Tree trunk
6	c1	Tree top
7	r6	Wagon
8	c2	Wheel left
9	c3	Wheel right

- (i) Describe the effect of the following statement:

1

```
list[4].setFillColour("red")
```

- (ii) Explain why the following statement generates an error:

2

```
list[8].getCircumference()
```

[Turn over

## 5. (continued)

- (d) The statement `sort(list)` in the main program invokes the procedure `sort()` and passes the actual parameter `list`. This procedure makes use of the bubble sort algorithm to arrange the contents of the variable `list` into descending order of area.

The design and data flow for this procedure is shown below.

- |  |                  |
|--|------------------|
| <ol style="list-style-type: none"> <li>1. procedure <code>sort(array)</code></li> <li>2.     set <code>n</code> = number of shapes in the array</li> <li>3.     set <code>swapped</code> = true</li> <li>4.     start while loop</li> <li>5.         set <code>swapped</code> = false</li> <li>6.         start fixed loop for <code>i</code> from 0 to <code>n-2</code></li> <li>7.             if area of <code>array(i)</code> and <code>array(i+1)</code> are in the wrong order</li> <li>8.                 swap contents of <code>array(i)</code> and <code>array(i+1)</code></li> <li>9.                 set <code>swapped</code> = true</li> <li>10.             end if</li> <li>11.         end fixed loop</li> <li>12.         set <code>n</code> = <code>n-1</code></li> <li>13.     end conditional loop</li> <li>14. end procedure</li> </ol> | <p>IN: array</p> |
|--|------------------|

Using a programming language of your choice, write code to implement the following steps of the design:

- |   |                            |
|---|----------------------------|
| <ol style="list-style-type: none"> <li>(i) Step 4</li> <li>(ii) Step 7</li> <li>(iii) Step 8</li> </ol> | <p>1</p> <p>2</p> <p>1</p> |
|---|----------------------------|