Practical Exercise 4 – House with Methods

The purpose of this task is to take the House program from Prac 1 and rewrite it, converting the code from the *paint()* method into a series of *methods* that will be called from *paint().*

A method is a named sequence of statements, that can be *called* from elsewhere in a program. A method enables code to be written once and used many times, including in other classes (e.g. *Math.random(), g.drawString(),* etc.). Methods can take multiple *arguments* and optionally return a value.

For example, in Prac 1 the following code was used in *paint()* to draw the house body:

// Draw the body of house

g.setColor(Color.blue);

g.fillRect(100,100,120,120);

This can be converted to a method called *drawBody()*. E.g.

public void drawBody(Graphics g)

{

// Draw the body of house

g.setColor(Color.blue);

g.fillRect(100,100,120,120);

}

This drawBody method would be called like this:

drawBody(g);

Better yet, we could include parameters for all the variables being used. E.g.

public void drawBody(int x, int y, int width, int height, Graphics g) // these are called “formal paramters”

{

// Draw the body of house

g.setColor(Color.blue);

g.fillRect(x, y, width, height);

}

This version of the drawBody() method could be called like this:

drawBody(100, 100, 120, 120, g); // these are called “actual parameters”

In practice, however, it probably wouldn’t be worth creating a method for just the body of the house. Instead it would make more sense to create a method for drawing the entire house, e.g. *drawHouse(x, y, g),* so that you could multiple houses by repeatedly calling the method with different parameters.

drawHouse(100, 400, g);

drawHouse(300, 400, g);

drawHouse(500, 400, g);

drawHouse(700, 400, g);

drawHouse(900, 400, g);

## Task 1: Break up your code into methods

Modify your code from Prac 1 so that you call a method to draw each element of the original scene (e.g. drawHouse(), drawSun, drawCloud, drawFence(), drawPath(), etc.).

For each method, include the appropriate parameters for all the variables being used.

Make use of this approach to draw multiple copies of a particular element in different positions/sizes within the scene.

e.g. make a drawWindow() method that will draw one window, then call this method twice to draw multiple windows.

## Task 2: Use a loop

Use a loop to repeatedly call one or more methods with different values each time. E.g. to draw a fence, a bunch of clouds, or a flock of birds.

Let’s say you have created a method *drawBird(int x, int y, Graphics g)* that draws a bird at (x, y). You could use the following code to draw multiple birds across the screen.

for(int xPos=100; xPos<=500; xPos = xPos + 200)

{

drawBird(xPos, 50, g);

}

## Task 3: Go further (stretch goal - bonus marks)

Add some special sauce to your code. For example, add one or more text fields or dropdown lists to allow the user to adjust the positions and colours of your drawing.

Or maybe you could pick something cool that you saw from someone else’s Prac 1 and incorporate it into your code.

Whatever, you do, make sure that you do it in a clever way, making good use of methods.

Aim to keep your code DRY (i.e. Don’t Repeat Yourself).