From Figure to Methods

SUMMARY:

As a junior in GNB Voc-Tech studying Programming and Web Development, I achieved a deep understanding in how to transition from an old project from sophomore year into something more convenient and simple for programmers to use. First of all, I analyzed the aged project and began to reference it into a method. Afterwards, I tinkered with parameters within methods to change factors of the stick figure without having to touch the project from inside the method. Finally as I got an understanding of the two concepts, I splintered the projects accessible to me into multiple methods to the point where I can call specific parts of the figure in their own methods. By understanding the logic of methods, it makes code a lot more reusable, flexible, maintainable, and readable.

BACKGROUND:

In order to get started, I needed to scrutinize the old project to get a basic idea of how the process of drawing the stick figure functioned. To begin, I searched for and opened up the project. Following this, I examined the code in order to refresh my mind with how I created the program about a year ago. Lastly, I conducted a couple of test cases to insure that the original code functioned properly.

MATERIALS:

In order to efficiently perform this, you will need to make sure that you have JRE (Java Runtime Environment) on your computer and some kind of IRE (integrated development environment) such as Eclipse or NetBeans to edit / run your code.

PROCEDURE:

The first of three procedures happened to be to simply place the stick figure code into a method and gain the power to reference it within the paint method. To start, I organized the skeleton for an applet (assuming you have a workspace and class developed for the code). Such processes include importing Applets before the class method and forging both an “init” method and a ‘paint” method (note to use graphics for the paint parameters). Next, I manufactured a new method within the main class with a name relevant to the task, for example “drawStickFigure”. Third, I copied and pasted part of the original stick figure program that drew the object into the

custom method produced within the newly made file. To finish it off, I generated a command in the paint method to reference the method “drawStickFigure” in its entirety. By following these steps, I received a basic understanding of how methods work and can be used within the coding environment.

After the completion of the previous project, I started a new one applying specific parameters into our methods. With this knowledge, I should be able to change the x and y coordinates of the entire stick figure by only changing two parameters from referencing my method. To begin the program, I first copied and pasted the previous class. Then, I edited the custom method (in this case “drawStickFigure”) to require two int parameters to be able to be function, which will eventually become the x and y of the stick figure. After that, I configured the code inside the method so that the coordinates of each part drawn refer to the two parameters given as the “point of origin”. Fourth, I manipulated the code that references my custom method by adding two integers (for example: 100 and 100). By gathering a deep understanding of both methods and parameters, the purpose of methods should become clearer in projects.

Now knowing how to use methods and parameters to improve my code, I could then delve deeper into the logic of methods and break down the stick figure. In order to break it down completely, I needed to make multiple projects and with each, break down the figure into smaller and simpler parts. First, I split my program from before to draw the head and body in their own separate methods, while still connecting to each other. Second, I broke down the head and body to different sub parts. The head separated into eyes, face, and mouth while the body divided into arms, legs and torso (since my project consisted of more than just a stick figure, I applied a method to draw the rest of the image). Third, I went even deeper by now referencing each eye with the same method and even each limb (arms and legs) with only one method to put my knowledge to the test. Having completed these accurately, this ensured that my understanding of what methods are used for and the whole logic behind them is completely scratched and noted in my mind.

RESULTS:

After the completion of each project, test cases were run in order to double check the precision of the code compared to the expected result I imagined. For most of these projects, the cases were fairly simple (the first project I ran the code to test it). I only needed to change the x coordinates and the y coordinates of the stick figure when referencing the method. With these

cases, I searched for two things in particular: ensuring the figure appeared and that all of the parts linked together despite the x and y given. With the knowledge bestowed upon me, this can lead me into understanding and coding projects in a much simpler way than I knew before. Thankfully, the results I expected almost perfectly correlated with what actually occurred.

CONCLUSION:

Fortunately, doing this succeeded and established my insight on methods. The objective of building such projects happened to be learning method flexibility and reusability. We can reference one method for many tasks, such as drawing limbs. Now we can simplify our project to the point that everything method performs a single action. Methods can make future projects such as calculators or even games and breeze to logically break down and program, no matter what the computer language may be.

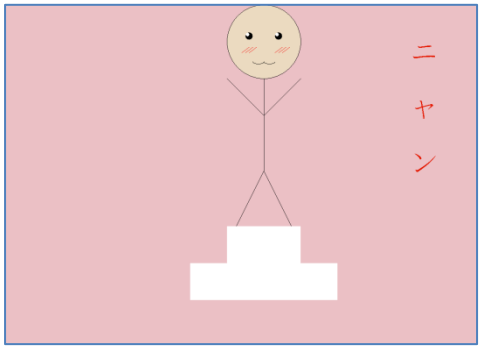

Figure 1: First project, drawing a stick figure with just one method.

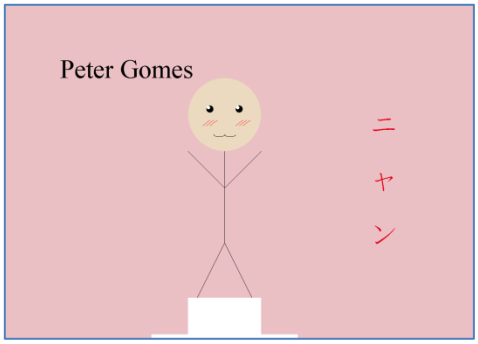

Figure 2: The stick figure project now enhanced with parameters to change the x and y.

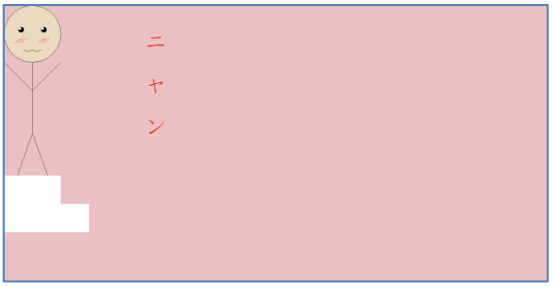

Figure 3: Drawing the same stick figure, but now drawing everything with its own method.

```
public class StickFigureMethod extends Applet {

	/*
	 * Author: Peter Gomes
	 * Date: 9/22/16
	 *
	 * Hello ^3^
	 */

	public void init(){
		//Custom Colors
		Color Pink = new Color (235, 192, 197);
		//End of Custom colors
		this.setBackground(Pink);
		this.setSize(800,800);

	}// end of init

	public void drawStickFigure(Graphics g){
		//Facial Structure
		Color Peach = new Color (235, 218, 192);
		g.setColor(Peach);
		g.fillOval(500, 200, 200,200);//Face
		g.setColor(Color.black);
		//Font Change
		Font f = new Font("serif", Font.PLAIN, 72);
		g.setFont(f);
		g.drawString("Peter Gomes", 150, 200);//Name of Author
		g.drawArc(570, 350, 30, 10, 170, 200);
		g.drawArc(600, 350, 30, 10, 10, -200);
		//Mouth
	//Blushies
		g.setColor(Color.red);
		g.drawLine(560, 315, 540, 330);
		g.drawLine(570, 315, 550, 330);
		g.drawLine(580, 315, 560, 330);
		g.drawLine(650, 315, 630, 330);//Start of right cheek
		g.drawLine(660, 315, 640, 330);
		g.drawLine(670, 315, 650, 330);
	//Eyes
		g.setColor(Color.black);
		g.fillOval(550, 275, 20,20);//Left
		g.fillOval(630, 275, 20,20);//Right
		//Shines of Eyes
		g.setColor(Color.white);
		g.fillOval(551, 276, 10,10);//Left
		g.fillOval(631, 276, 10,10);//Right
		g.setColor(Color.black);
	//Start of Body
		g.drawLine(600, 400, 600, 650);
	//Arms
		g.drawLine(600, 500, 500, 400);
		g.drawLine(600, 500, 700, 400);
	//Legs
		g.drawLine(600, 650, 525, 800);
		g.drawLine(600, 650, 675, 800);
	//Podium
		g.setColor(Color.white);
		g.fillRect(500, 800, 200, 100);
		g.fillRect(400, 900, 400, 100);
	//Hiragana
		Color DarkRed = new Color (230, 0, 23);
		g.setColor(DarkRed);
		g.drawString("ニ", 1000, 350);
		g.drawString("ャ", 1000, 500);//Nya
		g.drawString("ン", 1000, 650);//N
	}

	public void paint(Graphics g){
		drawStickFigure(g);
	}

}
```

Figure 4: The class for the first stick figure, using only one method to call the entire stick figure.