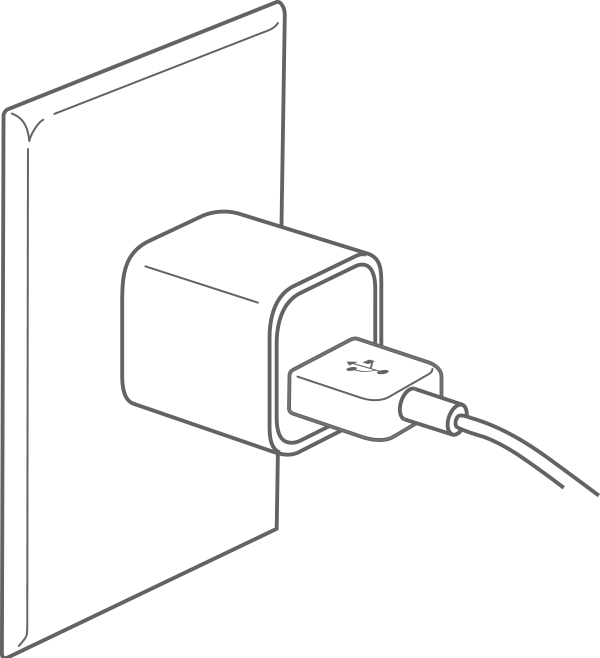
CS 477-51 Design Patterns Homework #1 January 17, 2019

Due: Thursday, January 24, 2019 Worth: 30 pts.

1. Here are the two variations of the first quiz. Answer the question you were not given in class.

Quiz A: USB charger example

To charge my cell phone, I use a USB cable and a standard US wall outlet. The USB cable does not fit directly in the outlet, so I use a little device (With a JCU logo on it!), that plugs into the wall and has a USB slot in it. Using this scenario as an example, explain the Adapter Design Pattern. Use a diagram if needed and be sure to identify the terms *client*, *target*, *adapter*, and *adaptee*.  
  
This is an Adapter Design Pattern because it implements the use of an *“adapter”* – the JCU logoed device – in order to allow two things to interact that would otherwise be unable to. The *“client,”* or who we are trying to adapt for is us – or anyone trying to charge their phone with a USB port. As such, our *“target”* is the USB cable. Since we are trying to plug the USB cable in the wall outlet, this would make the wall outlet the *“adaptee”* – or who you are adapting for.



2.) Suppose that you have an electronic thermometer that generates the current outdoor temperature in Fahrenheit. Suppose there is a website that displays the temperature digitally in Celsius.

You want to write a program that uses both resources, but you cannot modify either API.

You have one method get the current temperature (in degrees Fahrenheit) from the e-Thermometer:

**double getCurrentTemp();**

And you have another method to send the temperature (in degrees Celsius) to the website:

**void updateTempSite(double tempC);**

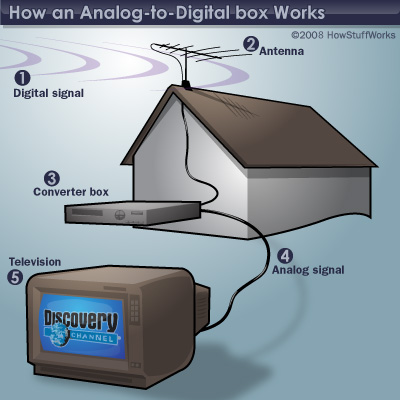
Recall that the conversion equations between Celsius and Fahrenheit are:

F = ((9/5)\*C)+32 and C = 5/9\*(F – 32)

Write an Adapter for the **updateTempSite** method so that it accepts the values that are directly provided by the **getCurrentTemp** method.

Quiz B: Digital TV example

On June 12, 2009 analog television signals stopped being broadcast; from that date forward, only digital television signals have filled the airways. People with cable boxes or cable-ready televisions could get the new digital signals, but other folks with older, analog televisions had to purchase a special piece of equipment called a converter box in order to keep watching. Converter boxes convert digital signals into analog signals for use with an analog TV. Use this scenario as an example to explain the adapter pattern. Use a diagram if needed and be sure to identify the terms *client*, *target*, *adapter*, and *adaptee*.



2.) Suppose that you have an electronic thermometer that generates the current outdoor temperature in Fahrenheit. Suppose there is a website that displays the temperature digitally in Celsius.

You want to write a program that uses both resources, but you cannot modify either API.

You have one method get the current temperature (in degrees Fahrenheit) from the e-Thermometer:

**double getCurrentTemp();**

And you have another method to send the temperature (in degrees Celsius) to the website:

**void updateTempSite(double tempC);**

Recall that the conversion equations between Celsius and Fahrenheit are:

F = ((9/5)\*C)+32 and C = 5/9\*(F – 32)

Write an Adapter for the **getCurrentTemp** method so that it returns a value that can be directly passed into the **updateTempSite** method.

**Question 2 – Part 1 – Also in .jar file:**

**public class TemperatureAdapter implements UpdateTempSite{**

**private FahrenheitThermometer fTherm;**

**public TemperatureAdapter()**

**{**

**fTherm = new FahrenheitThermometer();**

**}**

**public double convertToCelcius(double fahrenheit)**

**{**

**double celsius = 5/9\*(fahrenheit - 32);**

**}**

***@Override***

**public double updateTempSite()**

**{**

**//TODO auto-generated method stub.**

**double temp = fTherm.getCurrentTemp();**

**return convertToCelcius(temp);**

**}**

**}**

2.) Take the code provided in the .zip file and do the following tasks.. (Note you must actually write code that compiles and runs for this task:

* 1. Write a class for Egyptian Throwing Sticks (ETS)

There are 4 Egyptian Throwing Sticks in a group, one side is rounded and painted, the other is flat and not painted (see picture and diagram). Your class should have on method called throwSticks() that returns an array of size four, with a 1 if the painted side of the stick landed up and a 0 if the unpainted side of the stick landed up. For purposes of this assignment, we will assume that there is a 60% probability that a stick will land on the flat face (painted side up) and a 40% chance that it will land on the rounded side (unpainted side up). Your implementation of ETS must reflect these probabilities.



* 1. Write an ADAPTER class to include ETS as a way of generating random numbers. The ADAPTER class must translate the array into a random number according to this metric: 1 painted side up generates the number 1, 2 painted sides up generates a 2; 3 generate a 3; 4 generate a 4, and 0 generate a 5.Modify the GenRandom program to call this random device in a similar fashion to the others.  
     (*SEE .JAR FILE)*

3.) Consider a method genFunction() in which you pass in an x-coordinate and it computes the appropriate y-coordinate according to a provided function (for example: 5x2 -7x + 3 or 2sin(x) + 2cos(x) ). The function is written expecting the x-value from the standard Cartesian coordinate system. It produces the corresponding y-value for the same coordinate system. (In this system the origin (0,0) is at the center of the screen and the range for coordinates in both x and y is -100 to 100.

In Processing, the coordinate system has the origin in the Upper Lefthand corner, x increases from left to right and y increases from top to bottom. A typical size for the drawing canvas is 600 x 600 pixels.

Write an Adapter class that takes x-coordinates from Processing and produces y-coordinates for Processing by calling genFunction() with appropriately modified inputs and outputs. Write the Adapter class in Java, but this program does not have to compile since you do not have either the calling code or the genFunction() method.

To make this conversion, you will need to translate the coordinates from the upper left to the center, scale it from 0..600 to -100…100, and change the direction of the y-axis.

ANSWER TO QUESTION 3 BELOW *(Also in .jar file)*:

Steps:

1) Get x-coordinate from Processing

2) Plug x-coordinate into getFunction(), and save the result returned

3) Convert / Scale the returned y-coordinate from Cartesian to Processing

4) Return the y-coordinate in a method like getValue(int yCoordinate)

public class coordinateAdaptor{

int cartesianYCoordinate = 0;

int xCoordinate = 0;

public coordinateAdaptor(int xCoordinate)

{

this.xCoordinate = xCoordinate;

cartesianYCoordinate = getFunction(xCoordinate);

//rescaleYCoordinate(cartesianYCoordinate);

}

public int getValue()

{

int value = 0;

value = rescaleYCoordinate(cartesianYCoordinate);

return value;

}

public int rescaleYCoordinate(int cartesianYCoordinate)

{

int quadrant = getQuadrant(int xCoordinate);

//depending on the quadrant, implement switch statements,

//or possibly Strategy Pattern to convert and re-scale below

/\*

\* Translation and Scaling Code goes here!

\*

\*/

}

public int getQuadrant(int x)

{

//returns the quadrant the Cartesian quadrant is in

}

}