Exercise 1: Pounds and Pence (Algorithms & Random)

In England, instead of dollars and cents, people use pounds and pence. There are 100 pence to a pound, and people say “p” instead of pence (for example, “twenty p” instead of “twenty pence”). The following coins are currently in use: 1p, 2p, 5p, 10p, 20p, 50p, 1 pound, and 2 pounds.

For this exercise, you will figure out how to give someone a specified amount of money (between 1p and 5 pounds) using the fewest coins possible. For example, if the amount is 69p, you would give these coins: one 50p, one 10p, one 5p, and two 2p.

Part (a): Work with one or two others to design an algorithm that solves this problem. Write down your algorithm and trade algorithms with another group. Your team leader will give you some inputs to use with the algorithms. Execute the algorithm you have step by step, while the authors watch. If you think the other group is making a mistake with your algorithm, talk to them about it. If this exercise uncovers an error in your algorithm, fix it.

Part (b): Now take turns explaining your group’s algorithm (in English, not “Java speak”) to the rest of the group.

Part (c): And now use the laptops to implement a ChangeMaker class. The main method should first create a Random object, then loop 10 times. Each time around the loop it should do the following steps:

1. Use the Random object to get an integer between 1 and 500. That number is the amount of English money that you need to convert to change. For example, the number 104 means 1 pound and 4 pence. The number 500 means 5 pounds and no pence.

2. Use your algorithm to figure out what coins you would give for the random amount of money. Display the random integer first, then the coins in order from the largest coin (2 pounds) to the smallest (1 p).

For example, if the random number is 104, the output would be something like this:

104: one 1-pound coin, two 2-p coins

Remember that you can create a Random object with or without a seed. If you use a seed, the same sequence of “random” numbers will be generated each time you run your program. Which way should you create the Random object when you first write your code, and why?
Exercise 2: If and Switch Statements

Part (a): Divide into 3 groups. Each group choose one set of cards: yellow and pink, or blue and green, or gold and white. Each yellow/blue/gold card has a code fragment that includes an if statement, and each pink/green/white card has a code fragment that includes a switch statement. Your job is to match each if-fragment with the best equivalent switch-fragment. If there is more than one equivalent switch-fragment, the best one is the simplest (the one you’d prefer to see in an actual program).

Part (b): You should have some switch-fragments that are similar but not equivalent to any of the if-fragments. For each of these, explain why the switch-fragment isn’t equivalent to the similar if-fragment.

Exercise 3: If I only had a Rectangle... (More Programming)

We started the semester with an Artist class that gave us the following methods:

```java
void drawLineDown( int length )
void drawLineUp( int length )
void drawLineRight( int length )
void drawLineLeft( int length )
void moveRight( int d )
void moveLeft( int d )
void moveUp( int d )
void moveDown( int d )
```
**Part (a):** Now assume that you also have a method

```java
void drawRectangle( int width, int height )
```

that draws a rectangle with the given width and height, whose upper-left corner is at the current position (and after drawing the rectangle, the current position is still at that upper-left corner).

Write code that draws this picture (the dotted lines are just there to show you the grid of cells):

![Diagram of a 3x2 grid of cells with a 2x2 rectangle drawn inside.]

**Part (b):** Now write code that asks the user for the “dimensions” of the drawing -- how many squares across and how many down. (The example above has 3 across and 2 down.) Then your code should draw the correct picture.

**Part (c):** Now write the `drawRectangle` method itself, then work on the laptops to add the `drawRectangle` method to the `Artist` class. Then you can test the code you wrote for Part (b).

**Part (d):** If you want one more challenge, consider the pictures below, which show four `nested squares`.

![Nested squares of sizes 1 to 5.]

Write code that draws nested squares after asking the user for the size of the outermost square.
Exercise 4: Time is of the Essence! (Logical Thinking)

1. You’ve been lost in the African savanna for days and you’re starving. As luck would have it, you stumble upon a shack containing a fresh ostrich egg and a warning: Boil this egg for exactly 9 minutes!

   Fortunately there is a pot, water, salt, and a wood stove with supplies to start a fire, but you don’t have a timer, only two hourglasses. One measures four minutes and the other measures seven minutes.

   **Problem:** How can you cook the egg for exactly 9 minutes using only these two hourglasses?

2. You have two pieces of rope. You know that each one takes exactly one hour to burn. They are not necessarily of the same length or width as each other. They also are not of uniform width (for example, they may be wider in the middle than at the ends). Thus, burning half of the length of the rope does not necessarily take half an hour.

   **Problem:** By burning the ropes, how can you measure exactly 45 minutes worth of time?
if (x == 0) y = 1;
else if (x == 1) y = 2;
else y = 3;

if (x == 4) {
    x += 4;
} else if ((x > 4) && (x <= 7)) {
    x += 7;
}

if (x == 1 || x == 3) {
    System.out.println("Good Morning!");
} else {
    System.out.println("Good Afternoon!");
}

if (x == 0 && y == 0) doSomething();
else if (x == 0 && y == 1) doAnotherThing();
else doSomethingElse();

if (x > 2 && x < 4) x = 10;
if (x <= 0 || x > 4) x = 100;
switch (x) {
    case 0: y = 1;
        break;
    case 1: y = 2;
        break;
    default: y = 3;
}

y = 3;
switch (x) {
    case 0: y = 1;
}
switch (x) {
    case 1: y = 2;
}

switch (x) {
    case 0: y = 1;
    case 1: y = 2;
    default: y = 3;
}
switch (x) {
    case 4 : x = x + 4; break;
    case 5 :
    case 6 :
    case 7 : x = x + 7; break;
}

switch(x) {
    case 5: 
    case 6: 
    case 7: x = x + 7; 
          break;
    default: x = x + 4;
}

switch(x) {
    case 1:
    case 3: System.out.println("Good Morning!");
            break;
    default: System.out.println("Good Afternoon!");
}
Pink or Green or White Cards

switch(x) {
    case 1:
    case 3: System.out.println("Good Morning!");
}
System.out.println("Good Afternoon!");

switch (x) {
    case 0: switch (y) {
        case 0: doSomething();
                break;
        case 1: doAnotherThing();
                break;
        default:doSomethingElse();
    }
    break;
    default: doSomethingElse();
}
switch (x) {
    case 0: switch (y) {
        case 0: doSomething();
            break;
        case 1: doAnotherThing();
    }
    break;
    default: doSomethingElse();
}

switch (x) {
    case 0: if (y==0) doSomething();
        else doAnotherThing();
            break;
    default: doSomethingElse();
}
switch (x) {
    case 1:
    case 2:
    case 4: break;
    default : x = 100;
}

switch (x) {
    case 3: x = 10;
}

switch (x) {
    case 1:
    case 2:
    case 3:
    case 4: break;
    default: x = 100;
}
switch (x) {
  case 1:
  case 2:
  case 4: break;
  case 3: x = 10; break;
  default : x = 100;
}

switch (x) {
  case 3: x = 10; break;
  default : x = 100;
}