

LCS-ES Group Meeting #3

Exercise 1: Logical Thinking

This question isn't really about programming. However, it requires you to solve a problem by thinking about it logically, and that's a good skill for a programmer to have!

Assume that you have 8 coins, and you know that 7 are OK but one is bad. You know that the bad coin has a different weight than the good coins, but you don't know whether it's heavier or lighter.

Figure out how, using only a balance scale, you can find out which is the bad coin using just 3 weighings. Hint: Find a way to determine that half of the coins are OK with just 1 weighing.

Now do the same thing assuming that you have 9 coins, one of which is bad. (Still use just 3 weighings to find the bad coin.)

And now for a real challenge, do the same thing assuming that you have 13 coins.

Exercise 2: Using Objects and Designing Methods

For this exercise, we'll assume that someone has written an Artist class that provides the following methods to draw lines and circles:

`drawCircle(int diameter)`

This method draws a circle of the given diameter (in inches), centered around the current position (and doesn't change the current position).

`drawLineDown(int length)`

This method draws a vertical line of the given length, starting from the current position and going straight down. The current position is not changed.

`drawLineRight(int length)`

This method draws a horizontal line of the given length, starting from the current position and going straight to the right. The current position is not changed.

`moveRight(int d)`

This method moves the current position *d* inches to the right.

`moveLeft(int d)`

This method moves the current position *d* inches to the left.

`moveUp(int d)`

This method moves the current position *d* inches up.

`moveDown(int d)`

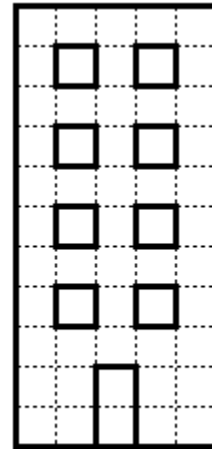
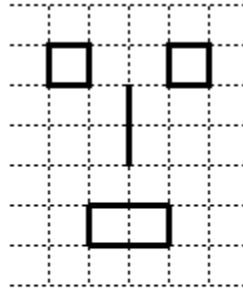
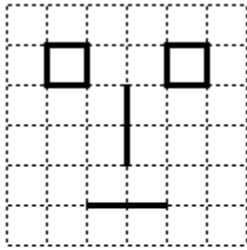
This method moves the current position *d* inches down.

Part (a). What is drawn when the following code executes?

```
Artist picasso = new Artist();

picasso.drawLineDown( 7 );
picasso.moveDown( 3 );
picasso.drawLineRight( 2 );
picasso.moveUp( 3 );
picasso.moveRight( 2 );
picasso.drawLineDown( 7 );
picasso.moveRight( 2 );
picasso.drawLineRight( 2 );
picasso.moveRight( 1 );
picasso.drawLineDown( 7 );
picasso.moveLeft( 1 );
picasso.moveDown( 7 );
picasso.drawLineRight( 2 );
```

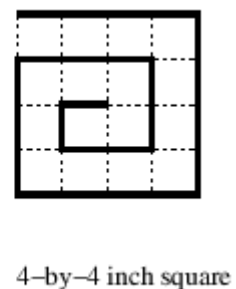
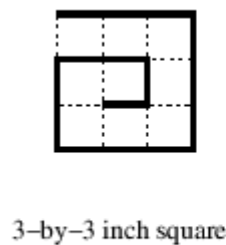
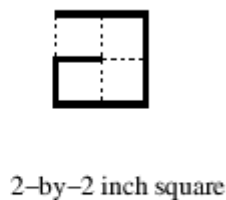
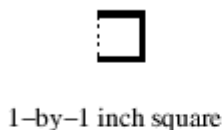
Part (b). For each of the pictures given below, write Java code that would make *picasso* draw the picture. (The dotted lines are not part of the pictures; they're there to show you how many inches you need to move or draw. Each dotted box is one inch on each side.)



Part (c). What other *Artist* methods would have made it easier to draw the pictures? Say what the methods' parameters should be, what the code in the methods should be, and how those methods would be used to simplify the code you wrote for Part (a).

Part (d). Most programming languages, including Java, let you write *loops*. For example, in Java you can essentially say "repeat the following statements *n* times", or you can say "repeat the following statements until *condition*", where you get to decide what the *condition* is. Can you simplify any of the code you wrote for Part (b) by using loops?

Part (e). Now look at the pictures below. Each is a *clockwise square spiral* that starts at the upper-left corner of the square, and stops when drawing the next line would run into a line that's already been drawn (again, the dotted lines are just there to show you a grid of inches).



What code could you use to make *picasso* draw a spiral in a three-by-three inch square?

In an *n*-by-*n* inch square?

Part (f). Can you write Java code that makes *picasso* write your name? If it isn't easy using just the *Artist* methods defined above, design your own methods.

Exercise 3: Designing Classes and Methods

Consider the following problem statement

Erin's Lemonade stand sells lemonade, grape koolaid, and ice water. Patrons can buy one or more large or small drinks per visit. Erin must buy lemonade mix, koolaid mix, and sugar from her Mom. For tax reasons Erin keeps a record of all sales, from which she can compute profits at the end of the day. She also uses this list to determine her best customers for special delivery service.

Your job is to design an object model for Erin. This typically involves four steps:

1. Identifying classes. A good source is the relevant nouns from the problem statement.
2. Identifying attributes. What state information is associated with each class you identified?
3. Identifying methods. What methods are associated with each class?
4. Identifying connections. What relations (if any) exist between your classes? These typically form “is-a” and “has-a” relations (*e.g.*, a car *is-a* vehicle and a dog *has-a* color).

For example, consider designing an object model for “Easy Pass”. Potential classes include car, truck, and toll. Potential data members include weight (of a truck), cost of a toll. Potential methods include payToll and connection might include a car is-a vehicle, cash is-a payment.

Draw simplified Unified Modeling Language (UML) diagrams to display your design. Recall that in UML

