CS390
Fundamental Programming Practices
Practice Problem

The code in your `prob1_old` package produces the following output when the `main` method in the `Driver` class is run:

```
/\ /\ \/ ||
```

Each of the figures displayed is represented by a separate class. For example, the figure

```
/\
```

is created by the class `HatMaker`. The main method in the `Driver` class creates an array of instances of these figure classes and then calls the constructor of `Driver`, which loops through the array and prints to the console each figure. However, to accomplish this, the code in the constructor tests the type of each figure class, and then downcasts the current object in the array to the correct type, and then calls the object's `getFigure` method in order to print the figure.

In this problem generalize these figure makers by creating an abstract class `Figure` with one abstract method `getFigure()`, and make all the figure maker classes be subclasses of `Figure`. In the constructor of your new version of `Driver`, you must use polymorphism to output figures instead of checking individual types. To make the instructions on this point clear, some of the new version of the Driver class has already been coded for you.

One other step for this exercise is to create on additional class, FaceMaker, whose `getFigure` method produces a figure like this:

```
:)
```

After you have finished coding your `main` method for the `Driver` class, the output after running the main method should look like this

```
/\ /\ \/ || :)
```

Here is a list of the things you need to do for this problem:

a. Create an abstract class `Figure` and have each of the other classes (other than `Driver`) `extend Figure` and place `Figure` and these modified classes in package `prob1_new`.
b. Modify the constructor of `Driver` so that polymorphism is used instead of checking individual types.
c. Create an additional class `FaceMaker` that produces this figure: `:)`
d. Recall that final output from the `main` method in `Driver` looks like this:
   `/\ /\ \/ || :)`