Instructions. There are no special instructions for these practice problems.

1. Consider the class called `Card` declared below. A Card object has a *denomination*: 2,3,...,10,J,Q,K,A and a *suit*: Hearts, Diamonds, Clubs and Spades. (‘J’ stands for ‘Jack’, ‘Q’ for ‘Queen’, ‘K’ for ‘King’ and ‘A’ for ‘Ace’). Let the denominations of Jacks, Queens, Kings and Aces be 11, 12, 13 and 14 respectively (in this way, all denominations can be represented by integers).

```java
class Card {
    private int denom;   //the denomination of a card
    private String suit; //the suit of a card

    //methods
}
```

(a) Write a constructor that creates a `Card` object with denomination `d` and suit `s`.
(b) Write a declaration that sets up a variable called `fourOfClubs` of type `Card` and initializes this variable to the 4 of clubs.
(c) Write methods called `getDenom` and `setSuit` as follows. The method `getDenom` returns the denomination of the calling card while `setSuit` sets the suit of the card to `s`.
(d) In many card games, there is a *point value* assigned to every card. If the denomination of a card is between 2 and 10, let its point value be its denomination. For example, the 4 of clubs is worth 4 points. Let the Jack, Queen and King of any suit be worth 10 points, and let the Ace be worth 11 points. Write a method called `computePointValue` that returns the point value of the calling card.
(e) Write a method called `printCard` that prints the calling card in the traditional manner, for example, ‘3 of Hearts’ or ‘Jack of Diamonds’.
(f) Now write a method called `isFlush` that returns `true` if the calling card along with the four cards passed as parameters is a *flush*. Otherwise, the method returns `false`. (Recall that a flush is a 5-card hand in which every card has the same suit.)

2. Consider a class called `DigitalTimer`. The objects of this class can be viewed as timers (or stopwatches, etc.) that record the time on a 24-hour (military-style) scale. For example, the time 15:34:00 represents the time 3:34p.m. Midnight is represented
by 00:00:00. Noon is represented by 12:00:00.

```java
class DigitalTimer {
    private int hours; //number of hours (between 0 and 23)
    private int minutes; //number of minutes (between 0 and 59)
    private int seconds; //number of seconds (between 0 and 59)

    //methods
}
```

(a) Write a method called `reset` that will set the calling object to midnight.

(b) Write a method called `add12Hours` that will advance the timer exactly 12 hours. Be careful!

(c) Write a method called `addOneSecond` that will increment the calling timer by exactly one second. Again, be careful! As an example of the problems to be faced, consider incrementing the time 23:59:59 by one second. The timer should go from 23:59:59 to 00:00:00.

(d) Write a method called `subtractOneSecond` that will decrement the calling timer by exactly one second. Watch out! Consider decrementing the time 00:00:00 by one second.

(e) Write a method called `compareTimes` that returns `true` if the calling timer has exactly the same time as does the timer `t` passed in as a parameter. If they have different times, the method returns `false`.

(f) Write a method called `cloneTimer` that creates and returns an exact duplicate of the calling timer.

3. Consider the class `Annulus` declared below. An annulus is shaped like a 'life saver', that is, a disk with a hole cut out of the center. It has an inner radius $r_{in}$ which is the radius of the inner disk (the disk that is cut out) and an outer radius $r_{out}$, the radius of the outer disk. Both inner and outer disks are centered at a point $(x, y)$. For an annulus to be properly formed, $r_{out} > r_{in} > 0$.

```java
class Annulus {
    private double radI, //radius of the inner disk
        radO, //radius of the outer disk
        xCen, //x coordinate of the center
        yCen; //y coordinate of the center

    //methods
}
```

(a) Write a constructor that creates an annulus having inner radius $r_{in}$, outer radius $r_{out}$ and is centered at the point $(x, y)$. The variables $r_{in}$, $r_{out}$, $x$ and $y$ are the formal parameters of the constructor. You may assume that $r_{in} < r_{out}$ so that the newly constructed annulus will be properly formed.
(b) Write a method called `area` that returns the area of the calling annulus. Recall that the area of a disk of radius $r$ is $\pi r^2$.

(c) Write a method called `diam` that returns the diameter of the calling annulus. The diameter of an annulus is defined to be the diameter of the outer disk.

(d) Write a method called `width` that returns the width of the calling annulus. The width of an annulus is defined to be the outer radius minus the inner radius.

(e) Write a method called `moveAnnulus` that translates the center of the calling annulus by $x$Dist in the horizontal dimension and $y$Dist in the vertical dimension. The variables $x$Dist and $y$Dist are the formal parameters. For example, if $x$Dist = 5 and $y$Dist = -3, the annulus is moved 5 units to the right and 3 units down.

(f) Write a method called `cloneAnnulus` that returns an exact copy of the calling annulus.

4. Use the class `Annulus` defined in problem 3 to write a complete Java program called `AnnulusDemo` that performs the following actions in the following order.

- Creates an annulus A with inner radius 3.25, outer radius 10.09 and center $(4.73, -2.11)$.
- Translates the annulus by $-4.12$ in the horizontal dimension and $8.11$ in the vertical dimension.
- Prints out its area, diameter and width.