Directions: Stave Game is an old Ohlone (Native American) game. A Stave is a two-sided thing (like a two-sided die or coin). Four staves are tossed all at once, then points are awarded as follows:

- If all land with the same side up, 2 points are awarded
- If exactly half land with the same side up, 1 point is awarded

After three plays, the game stops (technically, you play until one score is 5, but we'll simplify for now). **In class, we're creating only a one-player game that allows the user to play three times. All conditional structures are written for you.**

- Play Stave Game in class to get a feel for the flow of the game, data created, and decision making. ☺ Have fun!
- Make a paper diagram showing all classes and methods, as was modeled in class. Include arrows and the words "call," "return," and "parameter" on your diagram appropriately. Look at the directions below to know what methods should be included and in what class they occur.
- Copy the code skeleton from my website, then add in pseudocode before adding in Java code. Some parts of this are also referred to below, to make the directions more clear. **Be sure all methods and classes are well documented, with descriptions of how variables are being used.** Try to use words such as "parameter," "call," "method," "class," "return," "local," "field," and specific data types or processes.
- Add a testing plan to the top of the documentation for the one class that receives input from the user.
- Stave.java is the only program with main() of the three being used. From main(), it should call a method called playGame(), then scoreGame(). After scoreGame() runs, call updateScore(), which is also in Stave.java. Repeat these three times in the aforementioned order, so the player can play three times, before the game terminates.
- Be sure to instantiate the two other classes in Stave.java, so you can **call their methods with an instance argument**.
- Be sure all **field variables have access specifier of *private*** (taking advantage of Encapsulation).
- playGame() does this: Prompt a user to press enter, to roll. After reading this, rollStave () and displayStave() are called four times. This looks like this:

```
roll1= rollem.rollStave();
show.displayStave(roll1);
roll2= rollem.rollStave ();
show.displayStave(roll2);
roll3= rollem.rollStave ();
show.displayStave(roll3);
roll4= rollem.rollStave ();
show.displayStave(roll4);
```

  roll1, roll2, roll3, and roll4 are all **field** int variables to the Stave.java class.

- scoreGame() casts each of the four rolls as (char), then determines a score. **The field variable *score* should be initialized to zero.** The following code is in the body of scoreGame(); you should be able to follow some of it, but other parts we have not learned yet:

```
String rolls =  "";

rolls = "" + (char)roll1 + (char)roll2 + (char)roll3 + (char)roll4;   //cast the char to a String
```

```
int count = 0;

for (int index = 0; index<4; index ++)
{

        if (rolls.charAt(index) ==’$’)

                count ++;

}

if (count == 4 || count ==0)
        score += 2;
                else if (count == 2)
        score ++;
```

- updateScore() simply adds the score variable to a variable called runningScore, which is a field variable that began as zero. Save the addition of the old runningScore and score variables as runningScore. Then simply print runningScore.
- RollDice.java can be largely copied and modified to make it work for a new class called Roll.java. Roll.java has a method called rollStave(). It should be called on from Stave.java and return a value to Stave.java that is an integer value (35 or 36). This value should be made by using Math.random().
- ShowStave.java is a class. It has a method called displayStave() that is called on from Stave.java. It should use the value from Stave.java that rollStave generated, and print a character version of it; print output each time is either # ((char) 35) or $ ((char)36).
- **You’ll find that only main() needs static in its header. Don’t add static anywhere else please.**
- **To facilitate discussion and assistance, please do not introduce any additional control or iteration structures, even if you know how to do them. You should also not be changing the given code.**
    - **IF you finish early, you may save a second version and make changes to make the game run better.**