Problem Solving Session

- The remainder of today's class will comprise the ***problem solving session*** (***PSS***).
- Your instructor will divide you into ***teams of 3 or 4 students***.
- Each team will ***work together*** to solve the following problems over the course of ***20-30 minutes***.
  - You may work on paper, a white board, or digitally as determined by your instructor.
  - You will submit your solution by pushing it to GitHub before the end of class.
- Your instructor will go over the solution before the end of class.
- If there is any time remaining, you will begin work on your homework assignment.

Class participation is a significant part of your grade (20%). This includes in class activities and the problem solving session.

Your Course Assistants will grade your participation by verifying that you pushed your solutions before the end of the class period each day.

Problem Solving 1

Using the following problem statement, outline which classes you might need along with any attributes they contain.

You will not need a player or a location class.

Roulette is a game of chance based on a wheel with 37 numbered locations on it. As the wheel spins a ball is dropped onto it. The location where it lands is the winner. Each location has a number (0 - 36) and a color (red, black, or green). Bets are placed on where the ball will land. A player may bet on as many single numbers and/or colors (except green) as they would like. Winnings are then dispersed to a player for each correct choice.

```
public class Wheel {
      static final Color[] locations → has
      colors
      int winningLocation and getter
      Color winningColor() getter
      void spin() → sets winningLocation and
winningColor //if spin is in main, add setters
for attributes
}
public class Bet {
      Bet constructor (int betAmount, int
      betIndex)
      int betAmount with getter
      int betIndex with getter
}
public enum Color {
      RED,
      BLACK,
      NONE,
      GREEN;
}
```

```
public class Bet {
	private int betAmount;
	private int betIndex;
	private Color color;
	public Bet(int betAmount, int betIndex) {
		this.betAmount = betAmount;
		this.betIndex = betIndex;
		this.color = Color.NONE;
	}
	public Bet(int betAmount, Color color) {
		this.betAmount = betAmount;
		this.color = color;
		this.betIndex = -1;
	}
	public boolean isWinning(int index, Color
	color) {
		return index == this.betIndex ||
		color.equals(this.color);
	}
}
```

Problem Solving 2

The class used to store bets will need to be able to manage bets that are for a number or a color. How can you allow someone to easily create a bet for a number or a color without having to specify all 3 pieces of information each time?

Write the code for the a Bet class that uses your solution to the above question. You do not need to write accessors or mutators.

Problem Solving 3

Bets will be entered in the following format:
<1st Bet Amount> <1st Bet Location>, <2nd Bet Amount> <2nd Bet Location>, ...
Example:
`Place Bet: 200 12, 300 0, 150 22`

Assume you are given a string with all the bet information in it (everything after the ': ' in the previous example). Write the code necessary to create a `Bet` object using the information from the 1st bet in the string.

Split Usage:
`String.split (regex); // returns String[]`

You will need to use the bet constructor from the previous exercise. Assume all bets are for a number location (i.e. no colors)

```
String bets = "200 12, 300 0, 150 22";

// Code to create the first bet
String[] betsArray = bets.split(“,”);
Bet[] betObjects = new Bet[betsArray.length];

for(int i = 0; i < betsArray.length; i++) {
      String[] components = String.split(“ “);
      betObjects[i] = new
      Bet(Integer.valueOf(components[0]),
      Integer.valueOf(components[1]));
}

//this makes bets for entire output
```

```
// Variable to hold the ball's current location
private int ballLocation;
//import random if applicable
public void spin () {
      int rand_location = random.nextInt(0, 37);
      System.out.println(rand_location + " is the
winning number!");
      this.winningLocation = rand_location;
      this.winningColor =
      locations[rand_location];

}
```

Problem Solving 4

Write a `spin` method that displays the winning location number. The number must be randomly determined and be between 0 and 36 (inclusive). In addition to displaying the winning number, the method must also save its value into the `winningLocation` attribute.

Random Usage:

```
Random random = new Random();
random.nextInt(min, max); //Exclusive max
```