Object-Oriented Development

- In Object-Oriented development, the programmer reads the specification and selects objects to model the specification.
- Some objects will require new classes designed and written by the programmer
- Other objects will be created from existing classes
    - Classes previously written by the programmer, or other programmers
- **Reusability** is an important feature of object-oriented programming
    - Reduces development time
    - Decreases likelihood of bugs

Case Study – Object-Oriented Rock Paper Scissors

This Rock Paper Scissors game will be created using Object-Oriented development.

RPS2 Specifications: Played by one user, who "throws" either rock, paper or scissors. The game then randomly selects either rock, paper or scissors for the computer's throw. The winner is determined by comparing the two choices. The player can initially choose to play multiple rounds. At the end of the rounds, an overall winner is declared.

RPS2 Output Sketch:

```
How many rounds? 3
Enter your throw (1=ROCK, 2=PAPER, 3=SCISSORS): 3
You throw SCISSORS.
Computer throws PAPER.
You win!
Enter your throw (1=ROCK, 2=PAPER, 3=SCISSORS): 3
You throw SCISSORS.
Computer throws ROCK.
Computer wins!
Enter your throw (1=ROCK, 2=PAPER, 3=SCISSORS): 2
You throw PAPER.
Computer throws ROCK.
You win!
Computer wins 1.  You win 2.
You win!
```

ICS4U Module 4: Note ↓ Exercise 2c

RPS2 Algorithm

1. Prompt the player for the number of rounds.
2. For each round:
    Prompt for the player's throw.
    Generate the computer's throw.
    Announce the winner of the round.
3. Announce an overall winner.

RPS2 Code Design

- we will use a player object and a game object

Designs:

RPSPlayer

variables: playerName, playerThrow

methods:

makeThrow – prompts the player for throw

getThrow – returns the player's throw

RPSGame

variable: compThrow, playerWins, playerThrow

methods:

makeCompThrow – generates the computers throw

getCompThrow – returns the computers throw

announceWinner – displays a message indicating the throws and the winner.
Requires parameters for player throw and player name.

bigWinner – determine overall winner

ICS4U Module 4: Note ↓ Exercise 2c

Pseudocode

```
declare game object
declare player object

prompt player for number of rounds
for (i=0; i < rounds; i++) (
        prompt player for throw
        player.makeThrow(playerThrow);
        gameObject.makeThrow();
        gameObject.announceWinner(playerObject.getThrow);
)
gameObject.BigWinner
```

RPS2 Implementation

RPS2Player class

```
/**
 * models the player in a game of RPS
 */
public class RPSPlayer {
	private int playerThrow;
	//ROCK = 1, PAPER = 2, SCISSORS = 3

	/**
	 * constructor
	 * pre: none
	 * post: RPSPlayer object created. The player is given a default throw.
	 */
	public RPSPlayer() {
		playerThrow = 1;  //default throw
	}


	/**
	 * Sets the player's throw.
	 * pre: newThrow is the integer 1, 2, or 3.
	 * post: Player's throw has been made.
	 */
	 public void makeThrow(int newThrow){
		playerThrow = newThrow;
	 }


	/**
	 * Returns the player's throw.
	 * pre: none
	 * post: Player's throw has been returned.
	 */
	 public int getThrow() {
		return(playerThrow);
	 }

}
```

RPS2Game class

```
/**
 * Models a game of RPS
 */

import java.lang.Math;

public class RPSGame {
	public static final int ROCK = 1, PAPER = 2, SCISSORS = 3;
	private int compThrow;
	private int playerWins = 0, computerWins = 0;

	/**
	 * constructor
	 * pre: none
	 * post: RPSGame object created. Computer throw generated.
	 */
	public RPSGame() {
		compThrow = (int)(3 * Math.random() + 1); //1, 2, or 3
		playerWins = 0;
		computerWins = 0;
	}


	/**
	 * Computer's throw is generated (ROCK, PAPER, or SCISSORS)
	 * pre: none
	 * post: Computer's throw has been made.
	 */
	 public void makeCompThrow(){
		compThrow = (int)(3 * Math.random() + 1); //1, 2, or 3
	 }


	/**
	 * Returns the computer's throw.
	 * pre: none
	 * post: Computer's throw has been returned.
	 */
	public int getCompThrow() {
		return(compThrow);
	}


	/**
	 * Determines the winner of the round.
	 * pre: playerThrow is the integer 1, 2, or 3.
	 * post: Displays a message indicating throws. Compares player's
	 * throw to computer's throw and displays a message indicating
	 * the winner.
	 */
	 public void announceWinner(int playerThrow) {

		/* Inform player of throws */
		System.out.print("You throw ");
		switch (playerThrow) {
			case ROCK: System.out.println("ROCK."); break;
			case PAPER: System.out.println("PAPER."); break;
```

```
                case SCISSORS: System.out.println("SCISSORS."); break;
            }
            System.out.print("Computer throws ");
            switch (compThrow) {
                case ROCK: System.out.println("ROCK."); break;
                case PAPER: System.out.println("PAPER."); break;
                case SCISSORS: System.out.println("SCISSORS."); break;
            }

            /* Determine and annouce winner */
            if (playerThrow == ROCK && compThrow == ROCK) {
                System.out.println("It's a draw!");
            } else if (playerThrow == ROCK && compThrow == PAPER) {
                System.out.println("Computer wins!");
                computerWins += 1;
            } else if (playerThrow == ROCK && compThrow == SCISSORS) {
                System.out.println("You win!");
                playerWins += 1;
            }

            if (playerThrow == PAPER && compThrow == ROCK) {
                System.out.println("You win!");
                playerWins += 1;
            } else if (playerThrow == PAPER && compThrow == PAPER) {
                System.out.println("It's a draw!");
            } else if (playerThrow == PAPER && compThrow == SCISSORS) {
                System.out.println("Computer wins!");
                computerWins +=1;
            }

            if (playerThrow == SCISSORS && compThrow == ROCK) {
                System.out.println("Computer wins!");
                computerWins += 1;
            } else if (playerThrow == SCISSORS && compThrow == PAPER) {
                System.out.println("You win!");
                playerWins += 1;
            } else if (playerThrow == SCISSORS && compThrow == SCISSORS) {
                System.out.println("It's a draw!");
            }
        }


        /**
         * Displays the overall winner.
         * pre: none
         * post: Computer and player wins compared and
         * an overall winner announced.
         */
        public void bigWinner() {
            if (computerWins > playerWins){
                System.out.println("Computer wins!");
            } else if (playerWins > computerWins){
                System.out.println("You win!");
            } else {
                System.out.println("It's a draw!");
            }
        }

}
```

RPS2 client code

```
/*
 * RPS2.java from Module 3 (ICS4U)
 */

 import java.util.Scanner;

 /**
 * Computer plays Rock Paper Scissors against one player.
 */
 public class RPS2 {

      public static void main(String[] args) {
            RPSGame rps = new RPSGame();
            RPSPlayer rpsOpponent = new RPSPlayer();
            int rounds;
            int playerThrow;
            Scanner input = new Scanner(System.in);

            /* play RPS */
            System.out.print("How many rounds? ");
            rounds = input.nextInt();
            for (int i = 0; i < rounds; i++) {
                  System.out.print("Enter your throw (ROCK=1, PAPER=2,
SCISSORS=3): ");
                  playerThrow = input.nextInt();
                  rpsOpponent.makeThrow(playerThrow);

                  rps.makeCompThrow();
                  rps.announceWinner(rpsOpponent.getThrow());
            }
            rps.bigWinner();
      }
}
```

RPS2 Testing and Debugging

When a new class is written, client code should be written to test the class. For the RPSGame class, client code should test all the possible throw combinations.

Programming Exercise:

Modify the RPSPlayer class to include a `playerName` variable and methods named `assignName` and `getName`. The assignName() method has a String parameter `name` that is assigned to `playerName`. Modify the announceWinner() and bigWinner() methods in the RPSGame class to include a String parameter `name` that is the player's name. Change the message displayed in the announceWinner() and bigWinner() methods to include the player's name rather than the word "You". Modifying the two classes should produce output similar to the sketch:

```
Enter your name: Mrs. McDougall
How many rounds? 3

Enter your throw (1=ROCK, 2=PAPER, 3=SCISSORS): 3
Mrs. McDougall throws SCISSORS.
Computer throws PAPER.
Mrs. McDougall wins!

Enter your throw (1=ROCK, 2=PAPER, 3=SCISSORS): 3
Mrs. McDougall throws SCISSORS.
Computer throws ROCK.
Computer wins!

Enter your throw (1=ROCK, 2=PAPER, 3=SCISSORS): 2
Mrs. McDougall throws PAPER.
Computer throws ROCK.
Mrs. McDougall wins!

Computer wins 1.  Mrs. McDougall wins 2.
Mrs. McDougall wins!
```

Submit your entire project, including client code, to the **dropbox** for this exercise. Recall proper naming conventions for the outermost folder:

Last Name, First Name – Module 3, Exercise 2C – RPS2