Verification

Tic Tac Toe

Yesterday, we used methods to cut out repeated code. What three ORATE principles did we apply?

Reuse
Testing
Extensibility.

Pulling out repeated code helps with all three of these principles.

Today, we are going to use methods to help with the other two pieces of ORATE.

Organization.
Abstraction.

We are going to pull out code that has a distinct function and group it in a method.

These methods all verify if someone has won the game.

```
public boolean win()
```

Returns true if certain conditions are met

```
public boolean winRiddle (String word){
```

Returns true if the word matches the answer

```
public void rockPaperScissorsWin(char p1, char p2)
```

Toasts either player 1 or 2, based on who wins

```
public void win()
```

Toasts either X or O, based on who wins

What type is each kind of method?

A: No return, No Parameters	C: No return, Parameters
B: Return, No Parameters	D: Return, Parameters

1.
```
public boolean win()
```

2.
```
public boolean winRiddle (String word){
```

3.
```
public void rockPaperScissorsWin(char p1, char p2)
```

4.
```
public void win()
```

How do these methods help with organization?

As programs become more complex, having all the code inside the onCreate() function becomes impossible. A function is like a mini-program that we can code without having to think about the rest of the program. This allows us to reduce a complicated program into smaller, more manageable chunks, reducing the overall complexity.

1.
```
public boolean win()
```

2.
```
public boolean winRiddle (String word){
```

3.
```
public void rockPaperScissorsWin(char p1, char p2)
```

4.
```
public void win()
```

How do these methods help with organization?

1.
```
public boolean win()
```

2.
```
public boolean winRiddle (String word){
```

3.
```
public void rockPaperScissorsWin(char p1, char p2)
```

4.
```
public void win()
```

How do these methods help with abstraction?

1.
```
public boolean win()
```

2.
```
public boolean winRiddle (String word){
```

3.
```
public void rockPaperScissorsWin(char p1, char p2)
```

4.
```
public void win()
```

How do these methods help with abstraction?

In order to use a function, you only need to know its name, inputs and outputs. You don't need to know how it works. This lowers the amount of knowledge required to use other people's code, especially when using imports.

1.
```
public boolean win()
```

2.
```
public boolean winRiddle (String word){
```

3.
```
public void rockPaperScissorsWin(char p1, char p2)
```

4.
```
public void win()
```

Let's look at this method:

3. First, code the win methods. Then, using the method signatures, call the methods.

(a) Method:

```
public boolean win(){
        //returns true if 5 points (global variable), false otherwise
        if (points >= _______)
                return _________;
        else
                return __________;
}
```

Call:

```
_______________ ___________= ___________
if(_____________ == true)
  Toast.makeText(getApplicationContext(), "Win!",
Toast.LENGTH_SHORT).show();
```

3. First, code the win methods. Then, using the method signatures, call the methods.

(a) Method:

```
public boolean win(){
        //returns true if 5 points (global variable), false otherwise
        if (points >= ____5____)
                return ____true____;
        else
                return ____false____;
}
```

Call:

```
_____________ _____________= _____________
if(_____________ == true)
  Toast.makeText(getApplicationContext(), "Win!",
Toast.LENGTH_SHORT).show();
```

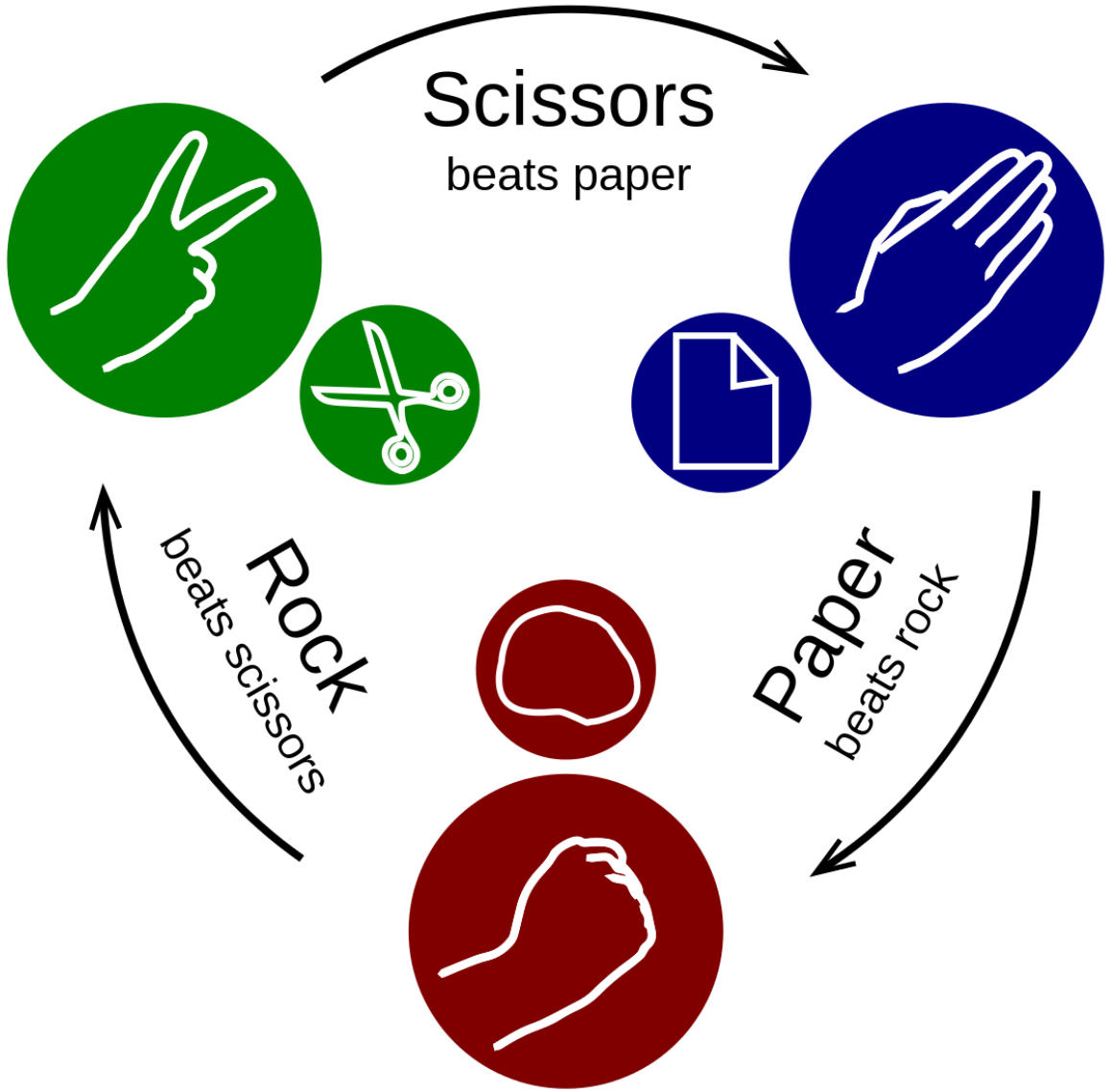
3. First, code the win methods. Then, using the method signatures, call the methods.

(a) Method:

```
public boolean win(){
        //returns true if 5 points (global variable), false otherwise
        if (points >= ____5____)
                return ___true____;
        else
                return ___false____;
}
```

Call:

```
_____boolean_____ ____w____= ____win();____
if(_____w_____ == true)
  Toast.makeText(getApplicationContext(), "Win!",
Toast.LENGTH_SHORT).show();
```



P1	P2	Winner
R	S	P1
S	P	P1
P	R	P1
R	R	Tie
S	S	Tie
P	P	Tie
R	P	P2
S	R	P2
P	S	P2

4. This method takes two chars, one for each player and using Rock, Paper, Scissors rules, displays a message for the winner. Fill in the blanks to make the method functional.


```
public void rockPaperScissorsWin(char p1, char p2) {
    int winner = 2;
    if (p1 == 'r' && p2 == '____')
        winner = 1;
    else if (p1 == 's' && p2 == '____')
        winner = 1;
    else if (p1 == 'p' && p2 == '____')
        winner = 1;
    else if (p1 == p2)
        winner = 0;
    if (winner == ____) {
        Toast.makeText(getApplicationContext(), "Player 1 Wins", Toast.LENGTH_SHORT).show();
    } else if (winner == 2) {
        Toast.makeText(getApplicationContext(), "_______________", Toast.LENGTH_SHORT).show();
    } else {
        Toast.makeText(getApplicationContext(), "Tie", Toast.LENGTH_SHORT).show();
    }
}
```

4. This method takes two chars, one for each player and using Rock, Paper, Scissors for the winner. Fill in the blanks to make the method functional.

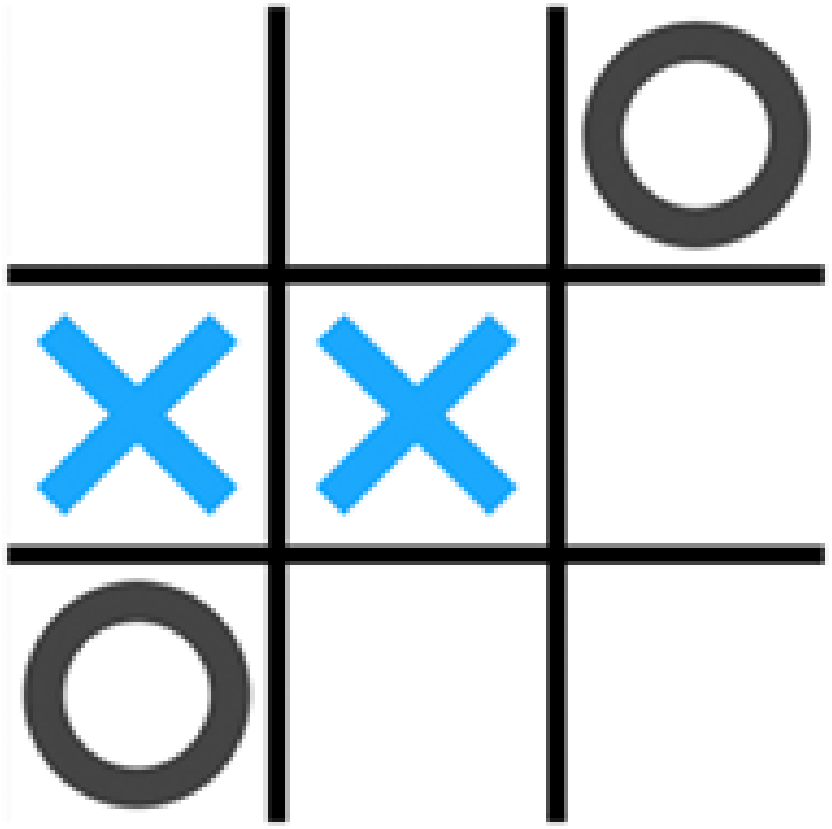
```
public void rockPaperScissorsWin(char p1, char p2) {
    int winner = 2;
    if (p1 == 'r' && p2 == '____')
        winner = 1;
    else if (p1 == 's' && p2 == '____')
        winner = 1;
    else if (p1 == 'p' && p2 == '____')
        winner = 1;
    else if (p1 == p2)
        winner = 0;
    if (winner == ____) {
        Toast.makeText(getApplicationContext(), "Player 1 Wins", Toast.
    } else if (winner == 2) {
        Toast.makeText(getApplicationContext(), "_______________", Toast
    } else {
        Toast.makeText(getApplicationContext(), "Tie", Toast.LENGTH_SHC
    }
}
```

P1	P2	Winner
R	S	P1
S	P	P1
P	R	P1
R	R	Tie
S	S	Tie
P	P	Tie
R	P	P2
S	R	P2
P	S	P2

4. This method takes two chars, one for each player and using Rock, Paper, Scissors for the winner. Fill in the blanks to make the method functional.

```
public void rockPaperScissorsWin(char p1, char p2) {
    int winner = 2;
    if (p1 == 'r' && p2 == '  s  ')
        winner = 1;
    else if (p1 == 's' && p2 == '  p  ')
        winner = 1;
    else if (p1 == 'p' && p2 == '  r  ')
        winner = 1;
    else if (p1 == p2)
        winner = 0;
    if (winner ==  1  ) {
        Toast.makeText(getApplicationContext(), "Player 1 Wins", Toast.
    } else if (winner == 2) {
        Toast.makeText(getApplicationContext(), "Player 2 Wins", Toast
    } else {
        Toast.makeText(getApplicationContext(), "Tie", Toast.LENGTH_SHC
    }
}
```

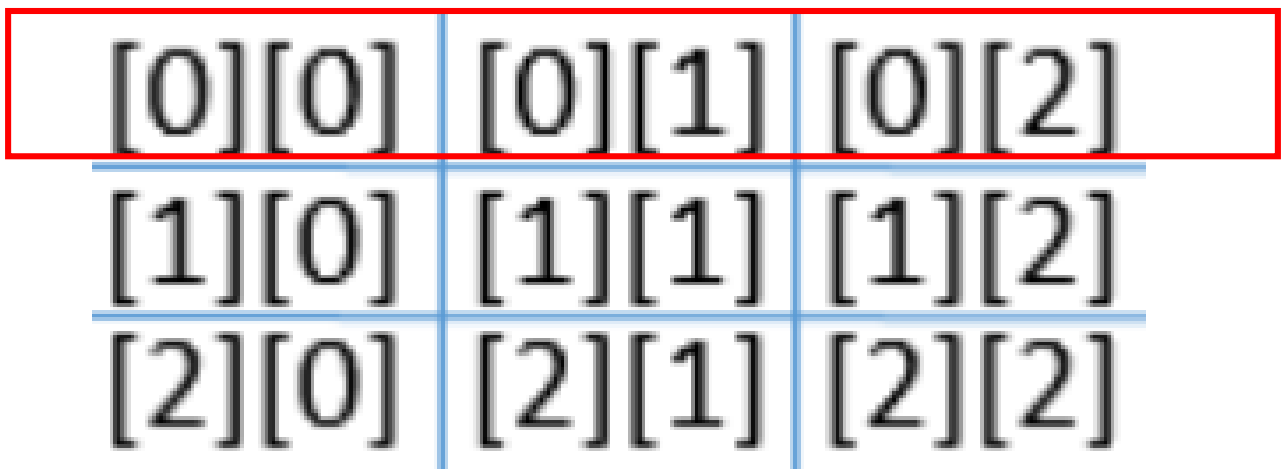
P1	P2	Winner
R	S	P1
S	P	P1
P	R	P1
R	R	Tie
S	S	Tie
P	P	Tie
R	P	P2
S	R	P2
P	S	P2



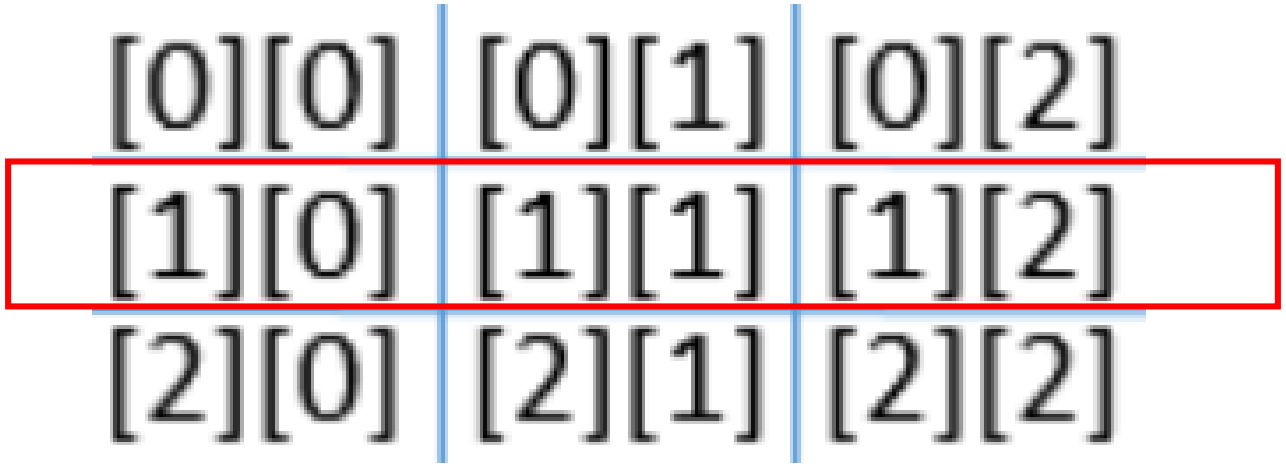
Tic Tac Toe
Win

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]

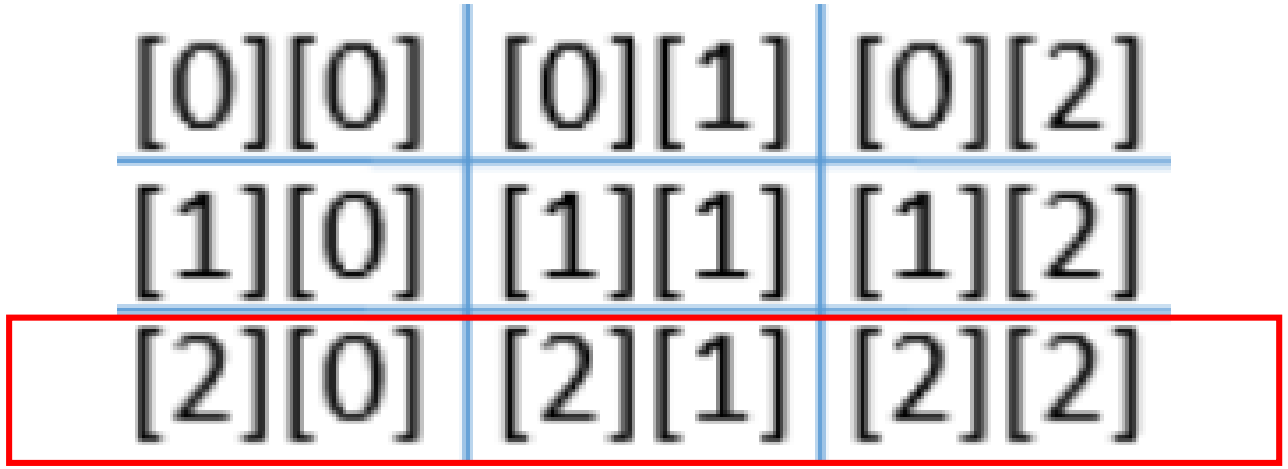
```
//board [x][y] holds 0 if empty; 1 if X holds square, 2 if O holds square
int winner = 0;
if (board[0][0] == board[0][1] && board[0][0] == board[0][2] && board[0][0] != 0)
    winner = board[0][0];
else if (board[1][0] == board[1][1] && board[__][__] == board[1][2] && board[1][0] != 0)
    winner = board[1][0];
else if (board[2][0] == board[2][1] && board[__][__] == board[2][2] && board[__][__] != 0)
    winner = board[2][0];
```



```
//board [x][y] holds 0 if empty; 1 if X holds square, 2 if O holds square
int winner = 0;
if (board[0][0] == board[0][1] && board[0][0] == board[0][2] && board[0][0] != 0)
    winner = board[0][0];
else if (board[1][0] == board[1][1] && board[1][0] == board[1][2] && board[1][0] != 0)
    winner = board[1][0];
else if (board[2][0] == board[2][1] && board[__][__] == board[2][2] && board[__][__] != 0)
    winner = board[2][0];
```



```
//board [x][y] holds 0 if empty; 1 if X holds square, 2 if O holds square
int winner = 0;
if (board[0][0] == board[0][1] && board[0][0] == board[0][2] && board[0][0] != 0)
    winner = board[0][0];
else if (board[1][0] == board[1][1] && board[1][0] == board[1][2] && board[1][0] != 0)
    winner = board[1][0];
else if (board[2][0] == board[2][1] && board[2][0] == board[2][2] && board[2][0] != 0)
    winner = board[2][0];
```



```
public void win() {
    //board [x][y] holds 0 if empty; 1 if X holds square, 2 if O holds square
    int winner = 0;
    if (board[0][0] == board[0][1] && board[0][0] == board[0][2] && board[0][0] != 0)
        winner = board[0][0];
    else if (board[1][0] == board[1][1] && board[__][__] == board[1][2] && board[1][0] != 0)
        winner = board[1][0];
    else if (board[2][0] == board[2][1] && board[__][__] == board[2][2] && board[__][__] != 0)
        winner = board[2][0];
    else if (board[0][0] == board[1][0] && board[__][__] == board[2][0] && board[__][__] != 0)
        winner = board[0][0];
    else if (board[0][1] == board[1][1] && board[__][__] == board[2][1] && board[__][__] != 0)
        __________ = _________[___][___];
    else if (board[0][2] == board[1][2] && board[__][__] == board[2][2] && board[__][__] != 0)
        winner = board[0][2];
    ______ __ (board[0][0] == board[1][1] && board[1][1] == board[__][__] && board[1][1] != 0)
        winner = board[1][1];
    else if (board[0][2] == board[1][1] && board[1][1] == board[__][__] && board[0][2] != 0)
        winner = board[0][2];
        //cat's game
    else if (board[__][__] != 0 && board[__][__] != 0 && board[__][__] != 0 &&
            board[1][0] != 0 && board[1][1] != 0 && board[1][2] != 0 &&
            board[__][__] != 0 && board[__][__] != 0 && board[__][__] != 0)
        winner = 3;

    if (winner == 1) {
        Toast.makeText(getApplicationContext(), "_____________", Toast.LENGTH_SHORT).show();
    } else if (winner == 2) {
        Toast._________(getApplicationContext(), "_____________", Toast.LENGTH_SHORT).show();
    } else if (_______________ == 3) {
        Toast.makeText(getApplicationContext(), "Cat's game", Toast._________________).show();
    }
}
```

```
public int win() {
    //board [x][y] holds 0 if empty; 1 if X holds square, 2 if O holds square
    int winner = 0;
    if (board[0][0] == board[0][1] && board[0][0] == board[0][2] && board[0][0] != 0)
        winner = board[0][0];
    else if (board[1][0] == board[1][1] && board[__][__] == board[1][2] && board[1][0] != 0)
        winner = board[1][0];
    else if (board[2][0] == board[2][1] && board[__][__] == board[2][2] && board[__][__] != 0)
        winner = board[2][0];
    else if (board[0][0] == board[1][0] && board[__][__] == board[2][0] && board[__][__] != 0)
        winner = board[0][0];
    else if (board[0][1] == board[1][1] && board[__][__] == board[2][1] && board[__][__] != 0)
        __________ = _________[___][___];
    else if (board[0][2] == board[1][2] && board[__][__] == board[2][2] && board[__][__] != 0)
        winner = board[0][2];
    ______ __ (board[0][0] == board[1][1] && board[1][1] == board[__][__] && board[1][1] != 0)
        winner = board[1][1];
    else if (board[0][2] == board[1][1] && board[1][1] == board[__][__] && board[0][2] != 0)
        winner = board[0][2];
        //cat's game
    else if (board[__][__] != 0 && board[__][__] != 0 && board[__][__] != 0 &&
            board[1][0] != 0 && board[1][1] != 0 && board[1][2] != 0 &&
            board[__][__] != 0 && board[__][__] != 0 && board[__][__] != 0)
        winner = 3;

    return winner;
}
```

Another way that I could have coded it

How would you call this?

What four values would be returned? What do they each mean?