

Functions



Slides courtesy of Dr. Abdallah Mohamed.

Announcements

- Bonus Test 3 is this week!
 Content: Variables, Images, Conditionals
- Reminder of the Labs/Activities 8-11 deadline
 - Will be due March 28th at 6 PM!
 - Come see me in my student hours (or make an appointment with me through Ed Discussion) if this will cause you problems ASAP so we can come up with an alternate plan

You must ALWAYS submit your pde files in your github repo and make sure any gifs you want marked are NOT in the `images` or the `animations` directories!

Objectives

- After these slides, you should be able to:
 - Recognize that functions are used to group statements that perform a particular task so that they can be easily used
 - Understand the difference between functions that return a value and those that do not.
 - Split your program into functions.
 - Create and use functions.



Programming Incrementally

- NEVER write code in a monolithic fashion. Instead, ALWAYS write code by adding only a few lines or features at a time and then testing.
- Thus, coding is an *incremental process*.
 - Write some code.
 - Test. Fix errors.
 - Repeat (until done).

Problem decomposition involves breaking down a large problem into subproblems which are easier to solve. Dividing problems into subproblems is called divide and conquer.

Problem Decomposition

- Breaking down a problem into subproblem makes it easier to solve the problem and also to track your code.
- Let's consider the Space Invaders video game. The steps inside the draw() method would be something like this:

void draw(){

- // Erase background.
- // Move spaceship based on user input.
- // Move enemies.
- // Move other game items (bullets).
- // Detect collisions between game items (e.g. bullets and enemies) and update game status accordingly.
- // Draw spaceship.
- // Draw enemies.
- // Draw bullets.



https://commons.wikimedia.org/wiki/File :Space_Invaders_style.png

Each one of the above steps requires many code lines. We can group relevant code lines into named blocks, i.e. functions.
cosc 123 – 5

Problem Decomposition, cont'd



void drawBulltes(){

Functions

- A function is a sequence of statements that performs a specific action.
 - Functions are also known as methods (in Java).
- Why do we create functions?
 - 1) To organize code into blocks that have specific purpose.
 - Each block of code is separated from other statements which makes it easier to read and modify (more *readability* and *maintainability*).
 - This makes the code more readable and easier to maintain
 - 2) To avoid duplication by reusing code.
 - The block of code can be called many times if the function needs to be done multiple times.
- What is the alternative?
 - Copy and paste and duplicate code. You will realize over time that this is actually the harder way to do things.

Built-In and Custom Functions

Two groups of functions:

1. Built-in Functions come with Processing

- You have already used many built-in functions such as:
 - Shape functions: rect(), line(), point(), ...
 - Vertex functions: vertex(), bezierVertex(), ...
 - Color functions: fill(), noFill(), stroke(), color(), ...
 - Image functions: image(), loadImage(),...
 - etc
- A full list of built-in functions with their description can be found at processing.org/reference
- 2. Custom functions
 - Can be added to your program either by:
 - 1) Creating them by yourself
 - 2) *Downloading* them in function libraries.
 - Useful libraries can be found at: processing.org/reference/libraries

Defining and Calling Functions

- Creating a function involves writing the statements in the function and providing a *function declaration* with:
 - a name (follows the same rules as identifiers)
 - list of the inputs (called parameters) and their data types
 - the output (return value) if any
- **Calling** (or executing) a function involves:
 - providing the name of the function
 - providing the values for all parameters (inputs) if any
 - providing space (variable name) to store the output (if any)

Defining a Function

Consider a method that converts a temperature in Celsius to Fahrenheit:



Calling Convert Function

ORDER OF OPERATIONS

____ float myFTemp; void draw() { myFTemp = convertC2F(50); print(myCTemp +"C is = "+myFTemp+"F"); 5 noLoop(); 6 send value 50 to 50 convertC2F as you call it float convertC2F(float tempInC) { 3 float f = tempInC / 5 * 9 + 32; //f=122 return f; <u>(122</u>) send result = 122 back to draw when you finish

COSC 123 - 11

Bouncing Ball Revisited

Without user-defined functions

```
float speedX = 1, speedY = 2;
float x=20, y=100, r = 20;
void setup(){size(200,200);}
void draw(){
    background(0);
```

//move ball

}

```
x += speedX;
y += speedY;
```

```
//detect collisions (edges)
if( x > width-r || x < r )
   speedX = -speedX;
if(y > height-r || y < r)
   speedY = -speedY;</pre>
```

```
//draw ball
ellipse(x,y,2*r,2*r);
```

WITH user-defined functions

```
float speedX = 1, speedY = 2;
float x=20, y=100, r = 20;
```

```
void setup(){size(200,200);}
```

```
void draw(){
   background(0);
   moveBall();
   checkCollisions();
   drawBall();
```

```
void moveBall(){
    x += speedX;
    y += speedY;
```

Using

functions

}

```
void checkCollisions(){
  if( x > width-r || x < r )
    speedX = -speedX;
  if(y > height-r || y < r)
    speedY = -speedY;</pre>
```

```
void drawBall(){
  ellipse(x,y,2*r,2*r);
```



```
COSC 123 – 12
```

Variables in Functions

Local variables

 Variables declared (created) in a function are local i.e. available only in that method.

Parameters

- As mentioned before, parameters are variables that allow for passing data into a function when calling it.
- Parameters are local variables that the function can use while it is executing.
- Each parameter has a data type.
- Functions may have zero parameters or as many as they want.
- To call a function with parameters you must pass in the necessary values (called arguments) for the function to use.
- Using parameters makes a function more powerful and useful.

Functions Notes

- When declaring a function, you must put the parenthesis "()" after the name even if the function has no parameters.
- If a function returns nothing, you can just say "return;".
- Parameter is the term used for input when viewing from inside the function (function's perspective). Argument is the term used for input when viewing from outside the function.
- Functions are declared only once, but can be called as many times as you want.
- Execution of the function halts at the return statement and any value in the statement is passed back to the caller.
- You may have multiple return statements in a function, but only one will ever be executed for a given execution.

Creating a Function

This function is supposed to take two numbers as input and return their sum. What is wrong with it?

```
int addTwoNum(int num1) {
    int result = num1 + num2;
}
```

- A. The two numbers are not added together.
- B. The result of the addition is not returned back.
- C. Only one number to add is passed into the function.
- D. The name of the function is not correct.

Creating a Function (2)

We want to create a function that multiplies two numbers together. Which of these functions is correct?

```
multTwoNum(int num1, num2) {
        return num1 * num2;
    }
    int multTwoNum(int num1, int num2, int num3){
Β.
        return num1 * n2;
    }
    int multTwoNum(int num1, int num2) {
        int result = num1 * num2;
    }
    int multTwoNum(int num1, int num2) {
        return num1 * num2;
    }
```

COSC 123 - 16

Functions

What is the output of this code?

```
int num=9;
void setup() {
  int result = doubleNum(num);
  print(result);
  noLoop();
}
int doubleNum(int n){
  return n*2;
}
```

A. nothing

B. error

C. 9

D. 18

Functions

What is the output of this code?

```
int num=9;
void draw() {
  int result = doubleNum(doubleNum(num));
  print(result);
  noLoop();
}
int doubleNum(int n){
  return n*2;
}
```

A. 36

B. 18

C. 9



Functions

What is the output of this code?

```
A. error
B. 3
C. -3
```

D. 0

```
int subtractNum(int a, int b) {
  return a-b;
}
void draw() {
  int x=5, y=8;
  int result = subtractNum(x, y);
  print(result + subtractNum(y, x));
  noLoop();
}
```

COSC 123 - 19

Functions

What is the output of this code?

A. oddodd

B. oddeven

C. evenodd

D. Eveneven

```
void setup() {
  int num = 10;
  print(evenOrOdd(11));
  print(evenOrOdd(num));
}
String evenOrOdd(int n) {
 if (n % 2 == 0)
   return "even";
 else
   return "odd";
}
```

Drawing a Spaceship

- Let's say we want to draw the spaceship in the figure.
- This code would do this task.



```
size(200, 200); background(255);
int x = 100, y = 100, size = 64; //location & size
// draw side guns
rectMode(CENTER);
stroke(255,90,90);
strokeWeight(1);
fill(255,0,0);
rect(x-size/3,y+size/2,size/15,size/3);
rect(x+size/3,y+size/2,size/15,size/3);
// draw jet engine
fill(255,180,0);
rect(x,y+size,size/2,size/10);
// draw main body
stroke(0);
fill(0,50,155);
triangle(x,y, x+size/2, y+size, x-size/2, y+size);
fill(0,100,255);
ellipse(x,y+2*size/3,size/3,size/2);
```

Drawing Two Spaceships

- Now let's say we want to draw another spaceship. You have two options:
 <u>Copy/paste the</u>
 - code again.
 - This works, but it will be ugly!
 - Use Functions!!



```
void draw() {
drawSpaceship(100,100,64);
drawSpaceship(50,50,32);
}
void drawSpaceship(int x, int y, int size) {
// draw side guns
rectMode(CENTER);
 stroke(255, 90, 90);
strokeWeight(1);
fill(255, 0, 0);
rect(x-size/3, y+size/2, size/15, size/3);
rect(x+size/3, y+size/2, size/15, size/3);
// draw jet engine
fill(255, 180, 0);
rect(x, y+size, size/2, size/10);
// draw main body
 stroke(0);
fill(0, 50, 155);
triangle(x,y,x+size/2,y+size,x-size/2, y+size);
fill(0, 100, 255);
ellipse(x, y+2*size/3, size/3, size/2);
}
```

Drawing Many Spaceships

Once you create the function, you can call it as many times as you wish, and each time you can change the parameters so that new output is produced. For example, the code below uses the drawSpaceship() function and produces the figure shown.

```
void setup(){
    size(300,250);
}
void draw() {
    background(255);
    drawSpaceship(100,100,64);
    drawSpaceship(250,40,32);
    drawSpaceship(50,50,32);
    drawSpaceship(150,80,32);
    drawSpaceship(150,80,32);
    drawSpaceship(200,100,96);
}
void drawSpaceship(int x,int y,int size){
    ...
```



End of Tuesday's Class



Functions



Slides courtesy of Dr. Abdallah Mohamed.

Announcements

- Bonus Test 3 is this week!
 Content: Variables, Images, Conditionals
- Reminder of the Labs/Activities 8-11 deadline
 - Will be due March 28th at 6 PM!
 - Come see me in my student hours (or make an appointment with me through Ed Discussion) if this will cause you problems ASAP so we can come up with an alternate plan
- You must ALWAYS submit your pde files in your github repo and make sure any gifs you want marked are NOT in the `images` or the `animations` directories!
- Final Exam Details

COSC 123 Final Exam

- Check SSC for the official date and time of the Final Exam
- There will be a mix of multiple choice questions (similar to Tests), but a <u>large</u> portion will be coding tasks, earning partial credit will be possible
- The final exam will be:
 Cumulative (Weeks 1-13)
 - Live (2.5 hours), will be invigilated (final details TBD)
 - Open book, open-notes, open-web but no cheating sites like Chegg/Course-Hero/Bartleby etc
 - Using IDEs (particularly PDE) is encouraged!
 - Combination of PrairieLearn, Gradescope and GitHub

COSC 123 Final Exam

- Thursday Apr 21 at 12:00 (Check SSC to confirm!)
- Online and invigilated by Zoom
- **Time:** 2.5 hours
- The exam covers all course material as indicated in the syllabus and during the lectures.
 - You will NOT be formally tested on Git and Command Line, but you will need those concepts to do the final exam (i.e. accept a GitHub Classroom repository)

Course Withdrawal Deadline is **March 18, 2022**

Passing requirements

- All students must satisfy ALL conditions to pass the course:
 - 1. Obtain an average grade of at least 50% on the Labs,
 - 2. Obtain an average grade of at least 50% on the Test and Exam components together,
 - 3. Obtain an average grade of at least 40% on the Final Exam,
 - 4. Obtain a grade of at least 50% on the whole course.

If students do not satisfy the appropriate requirements, the student will be assigned the **lower** of their earned course grade or, a maximum overall grade of 45% in the course.

Nice Ideas with Functions

Remember: Problem Decomposition

 Remember that you can use functions to break down your problem into reusable pieces of code.

```
float x = 100, y = 100, r = 20;
float speedX = 2, speedY = 3;
void draw() {
  background(0);
  x += speedX; y += speedY;
  if (x>width-r ||x<r)
      speedX=-speedX;
  if (y>height-r||y<r)
      speedY=-speedY;
  ellipse(x, y, 2*r, 2*r);
}
```

```
float x = 100, y = 100, r = 20;
float speedX = 2, speedY = 3;
void draw() {
  moveBall();
  bounceBall();
  drawElements();
void moveBall() {
  x += speedX; y += speedY;
void bounceBall() {
  if (x>width-r | x<r)
        speedX=-speedX;
  if (y>height-r||y<r)</pre>
        speedY=-speedY;
void drawElements() {
  background(0);
  ellipse(x, y, 2*r, 2*r);
```

IDEA1: Game Loops (Problem Decomposition)

- As you saw in the Space Invaders game in the readings, there is a game-loop that can be used in many games use
- The game loop does the following in every frame:
 Move game items (also update their other attributes)
 - Detect collisions
 - Draw game elements

Note: the order may change depending on your code!

Space Invader's Game Loop

 As you read in the pre-class readings, the game loop for the Space Invader game looks like this

void draw(){ //draw() is the game loop // Erase background. // Move spaceship based on user input. // Move enemies. // Move enemies. // Move other game items (bullets). // Detect collisions between game items (e.g. bullets and enemies) and update game status accordingly. // Draw spaceship. // Draw enemies. // Draw bullets.



https://commons.wikimedia.org/wiki/File :Space_Invaders_style.png

Space Invader's Game Loop, cont'd



void drawBulltes(){

COSC 123 - 34

Space Invader's Game Loop, cont'd

```
void draw() {
  background(0);
  moveSpaceship();
  detectCollision();
  drawElements();
void moveSpaceship() {
  x += speedX; y += speedY;
void bounceBall() {
  //for each enemy
  // for each bullet
  // if(dist(enemy,bullet) < T)</pre>
          collision(enemy, bullet)
  //
}
void drawSpaceship() {
  ellipse(...);
  rect(...);
```



https://commons.wikimedia.org/wiki/File :Space_Invaders_style.png

IDEA2: Multiple Scenes

Having multiple scenes in your animation or game is possible as follows:

- Create multiple functions, scene0(), scene1(), etc. Each will "act" as the draw method for the corresponding scene.
- 2) Create a variable, e.g. int scn, that keeps track of which scene is currently active.
 - For example, when scn=0, then scene 0 is active.
- 3) Use switch statement within draw() to display the right scene based on the value of scn.

```
switch(scn){
    case 0: scene0(); break;
    case 1: scene1(); break;
    ...etc
}
```

Simple Two-Scene Animation

- In this example, we show how to build an animation with two scenes:
 - Scene 0: intro screen
 - Scene 1: bouncing ball



Simple Two-Scene Animation, cont'd



Medieval Tales Game

Another example of using scenes is in our Medieval Tales game. The game has five scenes:



The example on the next slide shows how the player character can move across the two "Game-playing" scenes.
 For simplicity, the example uses a ball (instead of the player) that moves horizontally across two scene.

Bouncing Ball version 3

- Let's say we have a ball moving horizontally across two scenes: scene0 and scene1.
- The ball starts in scene0, moving to the right towards scene1.
- The ball bounce off the right edge of scene1 and the left edge of scene0
- When the ball reaches the edge between the two scenes, the animation should show the scene to which the ball is heading.



COSC 123 – 40

Bouncing Ball version 3 (Cont'd)

The code shows a ball bouncing off the edges. For simplicity, the ball moves only horizontally.

<u>IDEA</u>

- The switch statement will choose which scene to draw.
- Within each scene (see next slides),
 if statements are used to bounce the ball off the right edges and to move across the scenes.

```
float speedX=3, x=20, y=100, r=20;
int scn = 0; //start in fist scene
```

```
void setup(){
```

```
size(200,200);
rectMode(CENTER);
fill(0);
strokeWeight(15);
```

```
}
```

```
void draw(){
   switch(scn){
     case 0: scene0(); break;
     case 1: scene1(); break;
   }
}
```

}

}

Bouncing Ball version 3 (Cont'd)

void scene0(){
 //Draw scene elements
 background(loadImage("background_0.png"));
 text("SCENE 0", width/2, 20);
 ellipse(x,y,2*r,2*r);
 line(0,50,0, height);



}

}

Bouncing Ball version 3 (Cont'd)

void scene1(){ //Draw scene elements background(loadImage("background_1.png")); text("SCENE 1", width/2, 20); ellipse(x,y,2*r,2*r); line(width,50,width, height);

//Move ball and check edges x += speedX; if(x >= width-r) // if ball at right edge speedX = -speedX; // reverse direction else if(x<0){ // if ball at left edge scn = 0; // go to scene 0 x = width; // put the ball on the right of scene 0</pre>



Tint your Spaceship!

- Modify drawSpaceship() function given before so that you can also control the brightness of the spaceship. For example, the following code should produce the output given below.
 E.g. brightness should be given as a number in the range of 0..1:
 - 0 means extremely dark (black silhouette), 1 means normal colors
 - How: multiply the brightness value by every color component in your function, e.g. fill(255) should be changed to

drawSpaceship(20,30,16,0); //tiny spaceship
drawSpaceship(50,50,32,0.4); //small spaceship
drawSpaceship(150,30,48,0.8); //medium spaceship
drawSpaceship(100,100,64,1.0); //big spaceship



Draw Your Character with Functions

 Update the code you wrote earlier for your character so that the character is drawn in a function. The parameters should be the *position*, the *scale*, and the *body and belt colors*. Use this function to draw 2 or 3 of your character. Use this header: drawSuperhero(int x,int y, float scale, color c1, color c2);
 Example:

drawSuperhero(200,200,2.0,color(19,0,205),color(0,255,255)); drawSuperhero(70,70,0.75,color(190,0,10),color(255,10,100));

HINT: whenever you transform the coordinates inside a function, it is better to restore the original coordinates before leaving the function. HOW can you do this?



COSC 123 - 45

Lecture Activity Task

Update your game

- 1. Download the starter code and unzip to your compl
- 2. Open any of the three files in Processing.
 - You will notice all three files appear in 3 tabs.
- **3.** Run the code (Ctrl+R) and observe the output.
 - This is the same "Player Jumping" exercise we did before.
- 4. Implement the following two requirements
 - REQ1 : go to scene1 tab and use functions to organize your code (see the REQ comments in scene1).
 - 2. REQ2: add one more scene, named scene0, in a new tab (press the triangle button beside scene1 tab to create a new tab). The new scene should be the game's opening screen with the message "Hit 'P' to play, and 'E' to exit. Hit 'M' during gameplay to return to this screen". You will need to update the keyReleased function with the following:
 - if(we are in scene 0 and player hits P) then go to scene 1 if(we are in scene 0 and player hits E) then exit() if(we are in scene 1 and player hits M) then go to scene 0



Lecture Activity Task

Update your game, Cont'd







Review / Practice



Slides courtesy of Dr. Abdallah Mohamed.

What You Learned so Far

- Primitive shapes, text, and color, coordinates transformation
- Active programs
 - setup(), draw() methods
 - Event methods (keyPressed, MousePressed, ...)
 - System variables (mouseX, pmouseX, key, keyCode, ...)
- Built-in functions:
 - Math, size, noLoop, …
 - Randomness: random, noise
 - map, norm, constrain,
- Variables and Images
- Conditionals
 - controlling items, buttons, bouncing attributes, gravity
- Loops
- Functions
 - Game loops, animations with multiple scenes

Let's build a simple game

- You have exactly
 - 1 spaceship
 - 1 enemy
 - 1 bullet at any time
- Spaceship moves horizontally with the mouse, but it doesn't move vertically at all.
- Enemy moves in a sine wave path
 Two full cycles
- You can shoot one bullet by mouse click – you can't shoot another one until bullet goes out of screen (top)
- If bullet hits enemy, score++.



Basic Idea

 We shall use our game loop with functions to Move/update items
 Detect collisions
 Draw items

- Each item will have its own attributes: x,y, size
- The bullet will always have the same x,y of the spaceship as long as it is not fired. Once fired, it will move upwards while keeping its x position fixed
 - We will need one more attribute (boolean) to record the state of the bullet, i.e. whether it is *active* or not.
- We will use the dist() function for collision detection.

```
float ship x = 0, ship y = 375;
float bullet x = 0, bullet y = 375; boolean bullet active = false;
float enemy x = 200, enemy y = 0, enemy size = 40, bullet len = 10;
int score = 0;
void setup(){size(400,400); stroke(255); strokeWeight(3);}
void draw(){
  background(100);
  text("Score: " + score, 20,20);
  //game loop
  moveSpaceship();
  moveBullet();
  moveEnemey();
  detectCollisions();
  displaySpaceship();
  displayBullet();
  displayEnemy();
}
void moveSpaceship(){
  ship x = mouseX;
                                             Change enemy y from a distance to
}
                                             an angle represented in terms of PI
void moveEnemey(){
  enemy y += 0.3;
  enemy x = 200+100*sin(map(enemy y, 0, height, 0, 8*PI));
```

```
void moveBullet(){
  if(bullet active){
    bullet_y-=6;
    if(bullet_y<0) bullet_active = false;</pre>
  }else{
    bullet x = ship_x;
    bullet_y = ship_y;
  }
void displaySpaceship(){
  triangle(ship_x,ship_y,ship_x+10,ship_y+20,ship_x-10,ship_y+20);
void displayBullet(){
  line(bullet_x, bullet_y, bullet_x, bullet_y+ bullet_len);
}
void displayEnemy(){
  ellipse(enemy_x, enemy_y, enemy_size, enemy_size);
}
void detectCollisions(){
  if(dist(bullet_x,bullet_y,enemy_x,enemy_y)<enemy_size/2){ //not accurate
       bullet active = false;
       score++;
void mouseReleased(){
  bullet active = true;
```

Better way to organize code – Use Tabs

Put all the attributes and methods for each game item in one tab

```
spaceshooter
                                                                    bullet
                                                                         enemy
                                                                                ship
          bullet
 spaceshooter
                enemy
                       ship
                                                         float enemyX, enemyY, enemySize=50;
int score = 0;
                                                         void displayEnemy(){
void setup(){
                                                           fill(255,0,255); noStroke();
  size(500,500);
                                                           ellipse(enemyX, enemyY, enemySize, enemySize);
  textSize(22);
}
void draw(){
                                                         void moveEnemy(){
  background(0);
                                                           enemyY += 2;
  fill(255); text("Score: " +score, 50,50);
                                                           enemyX = 250 + 100*sin(map(enemyY,0,height,0,4*PI));
  moveShip();
                                                        10 // enemyX = 250 + 100*sin(enemyY/height * 4*PI);
  moveEnemy();
                                                           if(enemyY > height + enemySize/2) enemyY = -enemySize
  moveBullet();
  detectCollisions();
  displayShip();
                                                                       spaceshooter
                                                                                 bullet
                                                                                       enemy
                                                                                             ship 🔻
  displayEnemy();
                                                                      float bulletX, bulletY;
                                                                      boolean shooting = false;
  displayBullet();
                                                                      void displayBullet(){
                                                                         stroke(0,255,255); strokeWeight(3);
                                                                         line(bulletX, bulletY, bulletX, bulletY+10);
 spaceshooter
          bullet
                enemy
                      ship
float shipX, shipY=475;
                                                                      void moveBullet(){
void displayShip(){
                                                                         if(!shooting){
  fill(255); noStroke();
                                                                           bulletX = ship X;
  triangle(shipX,shipY, shipX+10,shipY+20, shipX-10,shipY+20);
                                                                           bulletY = shipY;
                                                                         }else{
void moveShip(){
                                                                           bulletY -= 15:
  shipX = mouseX;
                                                                           if(bulletY<-10) shooting = false;
```

Be Creative...

Expand the code from the previous slide to include any <u>one</u> of the following items. If you do more than one, that's even better!

- Improve game items (spaceships and bullet)
 - e.g. replace with images, use better vector designs (e.g. enemy should be more than just a moving dull circle), etc.
- Change the code so that the enemy follows one of three predefined paths every time it enters the window from the top. The chosen path should be selected randomly.
 - Define 3 paths: sinusoidal, noise(), and diagonal. Each time the enemy enters the screen, it randomly selects one of the 3 paths and follows it until either it exits the screen (from the bottom) or it is shot.
- Allow your spaceship to shoot another bullet (total of 2 bullets).
 - You cannot use arrays or OOP.

An Even Better way?

- The functions and attributes for each item are interleaved. This would make the code very hard to work with when we have many game items.
- Also, it is currently very tedious to name the variables and functions for each item in the game. This gets worse as we have more and more game items.
- Is there a better way to build this game??
 - Yes, by "grouping" all functions and attributes of each item under a code block, and then giving them simpler names.
 This can be done using Object Oriented Programming or



We will discuss this next week!