Lecture Notes

Print Statements

```
import static java.lang.System.out;

public class printing{
    public static void main(String[] args){
        out.println(“hello world”);
    }
}
```

Printing to the screen

-Main Method
-Maybe explaining to the experienced students what String[] args is
-Stressing case sensitiveness
-Semicolon
-format styles:
    indent each line so it looks neat
    keep lines not to long \
    // for single line comments
    /* */ for multiple line comments

Lesson 4 Variable Creation

-activity: bringing in boxes with different types of candies.explaining how a variable holds a kind of candy. either that or use prizes if parents are against the candy idea

-explain different kinds of variables to students.

Text Adventure(Typical Core Project):

Lesson 1 Starting an Adventure

Variable Types: People might struggle here with the diff between an int and double
Need to be able to explain what a decimal is

int
char
double
boolean
String

Variable Naming Convention
Printing Variables

Lesson 2 Enter the Cave

Arithmetic Operators Precedence:

Should show an example of bad code to show why () is neccasary

() Par

++ -- Increm

/ * % div, mult, mod

+ - addition subtracti

= += -= assignment
*= /= %=

Using operators in text adventure:
leveling up, dealing damage

Lesson 3 Using Input

scanner:

import java.util.Scanner //tell your translator to come

Scanner playerInput = new Scanner(System.in); //declare your translator

```
System.out.println("What is your name");
playerName = playerInput.nextLine();
playerName.close()
```

scanner methods:

nextLine	saves what the user types as a string
next	saves what a user types until the first space as a String
nextInt	saves a number that the user types
nextDouble	saves a decimal number

Lesson 4 If statements

if statement:

if statement checks the result of an expression if it is true

```
if (expression){
        // run if true
}else{
        // run if false
}
```

relation operators:

link operators with boolean expressions

==	equals
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
!=	not equal

string comparison

you cannot use relational operators to compare strings(checks bits)
.equals() which checks exact capitalization or .equalsIgnoreCase()

Lesson 5 Luck

Random Generator
import java.util.Random

random int from 0 to 10
randomGenerator.nextInt(10)

randomGenerator.nextBoolean()

CORE REFERENCE

Lesson 1 Variables and Data

Lesson 2 Arithmetic Oper

Lesson 3 Scanner

Lesson 4 If

Lesson 5 Random

PROGRAM FLOW

Lesson 1 Boolean Operators

allows you to check more than one condition

A && B	only true if and b are true
A || B	true if at least one is true
A ^ B	exclusive or. only true if one is true
!A	not. opposite of what A has

```
import java.util.Scanner;

public class BooleanOperators {

    public static void main(String[] args) {


        Scanner userInput = new Scanner(System.in);

        System.out.println("Choose a class:\n1 - Warrior\n2 -
Thief\n3 - Mage");

        int playerClass = userInput.nextInt();

        System.out.println("There is a guard here, what do you do?\n1
- Fight him\n2 - Steal his wallet\n3 - Throw a fireball\n4 - Talk the
guard into leaving\n5 - Intimidate the guard\n6 - Bond with the guard over
your shared disdain for magic");

        int playerChoice = userInput.nextInt();

        /* new code goes here */

        if( playerClass == 1 && playerChoice == 1){
              System.out.println("You use your awesome warrior skills
and punch the guard");
        }
        else if( playerClass == 2 && playerChoice == 2){
              System.out.println("You're so sneaky he doesn't notice
his wallet is gone for days.");
        }
        else if( playerClass == 3 && playerChoice == 3){
              System.out.println("You throw a very bright and shiny
fireball. The guard is slightly on fire.");
        }
        else {
              System.out.println("Your attempt fails and the guard
punches you.");
        }


        userInput.close();
    }

}
```

Lesson 2.1 While loops

while(expression){
code runs while expression is true
}
talk about continue and break statements

Lesson 2.2 For loops

```
//Syntax
for (initial value; condition; step){
//code
}
```

Lesson 2.3 For each loops

are used to loop through each data in an array

```
    public class ForEachExample {

    public static void main(String[] args) {

          String [] inventory = { "Sword", "Yellow Potion", "Tent",
"Potato Masher"};

          for( String inventoryItem : inventory){

              System.out.println(inventoryItem);

          }

    }

}
```

Lesson 3 Switch Statement

switch statement are like a bunch of if statements that checks an integer or char
break must be written

```
switch(choice){
case 1:
        //1 typed
case 2:
        //2 typed
case 3:
        //3
default:
        //run if user types any other number
}
```

for char case ‘b’

Lesson 4 Methods

access_modifier return_type name (type arguement, ….)

Lesson 5 Arrays

declaring:
String[] nameLst = {“s”,”b”,”a”};
or
String[] emptyList = new String[10];

numbering(must show 0):
accessing values:
emptyList[0]
length:
.length

Lesson 6 2D Arrays

draw a grid to explain this

int[][] array2D = new int[10][5]; 10 rows, 5 columns

```
for (int row = 0; row < map.length; row++){
            for(int col = 0; col < map[row].length; col++){
// Check each spot on the map and print the appropriate graphic
                        switch(map[row][col]){
                              case 0:
                                    // Water
                              System.out.print("~ ");
                                    break;
                              case 1:
                                    // Plains
                              System.out.print(". ");
                                    break;
                              case 2:
                                    // Mountain
                              System.out.print("^ ");
                                    break;
                                 }
                               }
                               // Line break
                               System.out.println();
                          }

                     }

}
```

Lesson 7 Exceptions

sometimes there are errors in code when an user enters an unexpected errors.

try and catch allows you to protect code that may cause an error.

```
try {
        //code that may break
}catch (Exception error){
        //fix the problem so the program can continue
}
```

throw allows you to throw an exception.

types of exceptions:

http://www.tutorialspoint.com/java/java_builtin_exceptions.htm

Examples:

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class ExceptionTest {

    static Scanner input = new Scanner(System.in);

    public static void main(String[] args) {

        float userMoney = 0;
        float userHealth = 0;

        System.out.println("How much money do you have?");
        try {
               userMoney = askForNumber();
        } catch (InputMismatchException e) {
               System.out.println("That isn't a number. You have 0 now.");
               userMoney = 0;
        }

        System.out.println("How much health do you have?");
        try {
               userHealth = askForNumber();
        } catch (InputMismatchException e) {
               System.out
                              .println("That's not a number, your health is set to
the default 20.");
               userHealth = 20;
        }

        System.out.println("You have " + userMoney + " money and " +userHealth +
" health.");
    }

    static float askForNumber() throws InputMismatchException {
        System.out.print("Enter a float: ");
        return input.nextFloat();
    }
```

Object Oriented Programming(Typical Intermediate Project):

Lesson 1: Classes & Lesson 2: Class Methods

```
1     public class Character {
2
3
4         public String name = "Geoff";
5
6         public int strength = 10;
7         public int health = 20;
8         public int defense = 5;
9
10        public int takeDamage(int damage){
11           int damageTaken = damage - this.defense;
12           this.health -= damageTaken;
             return damageTaken;
          }
          public int attack(Character target){
              int damageDealt = this.strength +
      Arena.generator.nextInt(5);
               return target.takeDamage(damageDealt);
          }
      }
```

```
1   import java.util.Random;
2
3   public class Arena {
4
5       // Create a random number generator for everyone to use. This is
6   easier than every character having their own.
7       // the "static" keyword means that there will only ever be one
8   of these.
9       public static Random generator = new Random();
10
11      public static void main(String[] args) {
12
13         Character player1 = new Character();
14         Character player2 = new Character();
15
16      }

    }
```

access an object by .
Lesson 3: Access Modifiers

the keyword public in front of classes, variables, and methods are access modifiers
public allows other classes to access data.

private: restricts use to same class file
protected: restricted to use within the same package or subclasses of the class

To access a private member variable you have to use getter and setter methods.

Packages: collection of classes
Organized Classes: you use to not need packages. you need to havea package identifier at the to.

think of it as being folders

Lesson 4: Constructors
a constructor is a method that tells java how to create an instance of a class. basically like the initiation for an object

Lesson 5: Extending a Class

Classes can inherit other classes. object inheritance is used with **extends** keyword

Character:

```
1   package Arena.Characters;
2
3   import Arena.Arena;
4
5
6
7   public class Character {
8
9       private static String[] nameList = {"Geoff", "Steve","Kruger" };
10
11      public String name;
12
13      // Private stats we don't want outside classes changing
```

```
14      public int strength = 10;
15      public int health = 20;
16      public int defense = 5;
17
18      public Character(){
19         // Class constructor
20
21         // Pick a character name at random
22         this.name =
23  nameList[Arena.generator.nextInt(nameList.length)];
24      }
25
26
27      public Character(int strength, int defense, int health){
28         this();
29         this.strength = strength;
30         this.defense = defense;
31         this.health = health;
32
33         }
34
35
36      // The code to run when this character attacks
37      public int attack(Character target){
38
39         // Apply damage formula
40         int damageDealt = this.strength;
41
42         // Tell the target to take this much damage, then return the
43  amount of damage the target took.
44         return target.takeDamage(damageDealt);
45
46      }
47
48      // The code to run when this character takes some damage
49      public int takeDamage(int damage){
50
51         // Apply defense formula
52         int damageTaken = damage - this.defense;
53
54         // Subtract the final damage number from this character's
55  health
56         this.health -= damageTaken;
57
58         return damageTaken;
```

```
 59      }
 60
 61      // This method will return the player's health so other classes
 62   can read it, but not change it.
 63      public int getHealth(){
 64         return this.health;
 65      }
 66
 67
 68      // This method checks if the character is alive
 69      public boolean isAlive(){
            return this.health > 0;
         }

     }
```

Areana.java

```
package Arena;

import java.util.Random;

import Arena.Characters.Character;


public class Arena {

    // Create a random number generator for everyone to use. This is
easier than every character having their own.
    // the "static" keyword means that there will only ever be one of
these.
    public static Random generator = new Random();

    public static void main(String[] args) {

       Character player1 = new Character();
       Character player2 = new Character(10,2,100);




       // Put a 2 after the name if characters are name the same
       if (player2.name.equals(player1.name)){
             player2.name += " 2";
       }

       System.out.println(player1.name + " vs. " + player2.name);
```

```
        // Fight as long as both characters are alive
        int turns = 0;
        while(player1.isAlive() && player2.isAlive()){
              turns++;

              System.out.println("Turn " + turns + "\n" + player1.name + ":
" + player1.getHealth() + " Health | " + player2.name + ": " +
player2.getHealth() + " Health\n");
              int damage;
              // player 1 attack
              damage = player1.attack(player2);
              System.out.println(player1.name + " hits " + player2.name + "
for " + damage + " damage.");

              // player 2 attack
              damage = player2.attack(player1);
              System.out.println(player2.name + " hits " + player1.name + "
for " + damage + " damage.\n");
        }

        // Check to see who won
        if( player1.isAlive()){
              System.out.println(player1.name + " wins!");
        }
        else if (player2.isAlive()){
              System.out.println(player2.name + " wins!");
        }
        else{
              System.out.println("It's a draw!");
        }

    }

}
```

Object Inheritance:

```
package Arena.Characters;

import Arena.Arena;

public class Rogue extends Character { //this is an addition to character

    // Rogue already has all the methods and variables of character - we
only need to change what will be different.
```

```
    public int dexterity = 20;


    public Rogue(){
       super(); //this means copy the character stuff
       this.strength = 8;
       this.defense = 4;

    }

    public int attack(Character target){ //overrides the other character
//                                          attack method

       boolean criticalHit = Arena.generator.nextInt(100) < dexterity;

       int damageDealt = this.strength +
Arena.generator.nextInt(this.strength) - this.strength / 2;
       if(criticalHit){
             damageDealt *= 2;
             System.out.println("*** Critical Hit ***");
       }
       return target.takeDamage(damageDealt);
    }

}
```

-teach inter about for each and 2d arrays
https://www.youtube.com/watch?v=vAZ8BJRaNkk

-germ wars is a proj for intermediate students

Germ Wars

Start The Project:

https://www.youtube.com/watch?v=0cqSjyvA8EY , https://processing.org/download/?processing download processing, create java project, import file system, then core.jar, add core.jar to build path

Processing

Setting up processing:

download processing from https://processing.org/download/?processing, then create a java project. Import a file system, and then go to that processing file you downloaded and import core.jar. Next right click core.jar to build the path to it.

or, create a new project with processing

Test Processing Was Downloaded Correctly

in public void setup(){

print “im working”

}

Processing Drawing

So processing is made mostly for creating java graphics. You can draw shapes instead of loading images for the project. The most common shapes are lines, rectangles, and elipses.

Draw a robot head:

STARTING

in setup or settings put size(800, 700); background(255, 255, 255);

for more colors visit: http://www.rapidtables.com/web/color/RGB_Color.htm

```
package processingdrawing;

import processing.core.PApplet;

public class ProcessingDrawing extends PApplet {

	public void setup()
	   {
	        size(800, 700);
	        background(255, 255, 255);
	   }
	   public void draw()
	   {
	   }
	   public void face(float xPos, float yPos){

 }
}
```

RECT and FILL

the fill method fills colors of drawn shapes. it takes 3 values for RGB
the rect method draws a rectangle. it takes x, y, width, and height

```
1     package processingdrawing;
2     import processing.core.PApplet;
3     public class ProcessingDrawing extends PApplet {
4
5           public void setup()
6              {
7                  size(800, 700);
8                  background(255, 255, 255);
9              }
10
11             public void draw()
12             {
13                 face(100,300);
14             }
15
16             public void face(float xPos, float
17    yPos){
18                 fill(0, 149, 185);
19                 rect(xPos, yPos, 500, 300);
20             }
21    }
22
23
```

The Eyes

The eclipse method draws an ellipse with parameters x,y, width, hieght

```
public void eyes(float eyesX, float eyesY)
      {
          fill(255, 255, 255);
          ellipse(eyesX, eyesY, 100, 100);
          ellipse(eyesX + 300, eyesY, 25, 25);
      }
```

The Mouth

the triangle method draws a traingle with 3 sets of x and y coordinates for each point of the triangle.

```
 public void mouth(float mouthX, float mouthY)
         {
             fill(255, 255, 255);
             triangle(mouthX, mouthY, mouthX +250, mouthY, mouthX + 100,
mouthY + 75);
         }
```

The Ears

They are rectangles (3) next to one another

first rectangle takes the points ax and ay

```
    public int pointAX = 75;
    public int pointAY = 400;

    public void leftEar()
    {
       fill(244, 121, 0); //orange
       rect(pointAX, pointAY, 25, 100); //first rectangle
       rect(pointAX - 25, pointAY + 12, 25, 75);//second rectangle
       line(pointAX - 35, 430, pointAX - 35, 100);//line
       rect(pointAX - 50, pointAY + 24, 25, 50); //third rectangle
       fill(255, 255, 255); //white
       ellipse(pointAX - 35, 100, 30, 30);
    }
```

DRAWING

to teach this get grid paper. it’s going to be a lot about geometry

```
1     // Expand this box to see code
2     // for the Right Ear
3
4     package screenspaceexperiements;
5
6     import processing.core.PApplet;
7
8     public class ScreenSpaceExperiements extends
9     PApplet
10    {
11
12        public void setup()
13        {
14           size(800, 700); //this takes width, height of
15    frame
16           background(255, 255, 255); //background
17    colors
18        }
19
20        public void draw() //automatically called
21    multiple times, need to call update if movement is
22    involved
23        {
24           face();
25           eyes();
26           mouth();
27           leftEar();
28           rightEar();
29        }
30
31        public void face()
32        {
33           fill(0, 149, 185);//fill in before drawing
34           rect(100, 300, 500, 300, 4); //rectangle
35        }
36
37        public void eyes()
38        {
39           fill(255, 255, 255); //fill in before drawing
40           ellipse(200, 400, 100, 100);// (200,400)
41           ellipse(500, 400, 25, 25); (500, 400)
```

```
42        }
43
44        public void mouth()
45        {
46           fill(255, 255, 255);
47           triangle(250, 500, 500, 500, 350, 575);
48        }
49
50        public int pointAX = 75;
51        public int pointAY = 400;
52
53        public void leftEar()
54        {
55           fill(244, 121, 0);
56           rect(pointAX, pointAY, 25, 100); //ear
57           rect(pointAX - 25, pointAY + 12, 25, 75);/ear
58           line(pointAX - 35, 430, pointAX - 35, 100);
59           rect(pointAX - 50, pointAY + 24, 25, 50);
60           fill(255, 255, 255); //white
61           ellipse(pointAX - 35, 100, 30, 30); //antena
62        }
63
64    public void rightEar()
65        {
66           fill(244, 121, 0);
67           rect(pointBX, pointBY, 25, 100);
68           rect(pointBX + 25, pointBY + 12, 25, 75);
69           line(pointBX + 65, 430, pointBX + 65, 100);
70           rect(pointBX + 50, pointBY + 24, 25, 50);
             fill(255, 255, 255);
             ellipse(pointBX + 65, 100, 30, 30);
        }
     }
```

MOVEMENT

```
package movement;

import processing.core.PApplet;
import processing.core.PImage;


public class Movement extends PApplet
{
    public float speed = 10;

    public float x = 400;
    public float y = 350;

    public boolean moveForward = false;

    public boolean moving = false;

    public void setup()
    {
       size(800, 700);
    }

    public void draw()
    {
       background(255, 255, 255);

       //Move and control the rocket ship.
       move();
       translate(x, y);
       drawRocketShip();
    }

    //Set up and draw the rocket ship.
    public int rocketX = 0;
    public int rocketY = 0;

    public void drawRocketShip()
    {
       //Body color.
       stroke(0, 149, 185);
       fill(0, 149, 185);
       //Body
       rect(rocketX, rocketY, 75, 50);
       //front
```

```
       triangle(rocketX + 75, rocketY + 1, 100, rocketY + 25, rocketX +
75, rocketY + 49);
       //window color
       fill(255, 255, 255);
       //window
       ellipse(rocketX + 60, rocketY + 25, 30, 15);
       //wings
       stroke(0, 149, 185);
       strokeWeight(3);
       fill(255, 255, 255);
       triangle(rocketX + 25, rocketY, rocketX - 15, rocketY - 25,
rocketX, rocketY);
       triangle(rocketX + 25, rocketY + 50, rocketX - 15, rocketY + 75,
rocketX, rocketY + 50);

       if(moving)
       {
             //Fire trail
             fill(255, 0, 0);
             noStroke();
             triangle(rocketX - 10, rocketY + 10, rocketX - 30, rocketY +
25, rocketX - 10, rocketY + 40);
       }
    }

    public void keyPressed()
    {
       if(key == 'w')
         {
             moveForward = true;
             moving = true;
         }
    }

    public void keyReleased()
    {
       if(key == 'w')
         {
             moveForward = false;
             moving = false;
         }
    }

    public void move()
    {
       if(moveForward)
       {
             x += speed;
```

```
            }
        }
    }
```

States

You can use states to create a menu, set times for players to jump. basically state is what should happen at a specific time

Using the mouse

```
package mouseinput;

import processing.core.PApplet;

public class MouseInput extends PApplet
{
	//position of the mouse
	float boxX = 350;
	float boxY = 250;

	//how big the mouse icon is
	int width = 100;
	int height = 100;

	//checks if the mouse is in the bound of the box
	boolean insideBox = false;


	boolean changeColor = false;

	int[] rgb = {105, 20, 155};

	public void settings()
	{
	    size(800, 600);
	}

	public void draw()
	{
	    background(0, 0, 0);
	    changeTheColor();

	    if(mouseX > boxX && mouseX < boxX + width
	            && mouseY > boxY && mouseY < boxY + height){
	        insideBox = true;
	    }
	    else{
```

```
            insideBox = false;
        }

          //colors the box
        fill(rgb[0], rgb[1], rgb[2]);
        rect(boxX, boxY, width, height);
    }

    public void mouseDragged()
    {
        if(insideBox)//if the mouse is inside the
        {
            boxX = mouseX - width/2;
            boxY = mouseY - height/2;
        }
    }

    public void mousePressed()
    {
        changeColor = true;
    }

    public void mouseReleased()
    {
        changeColor = false;
    }

    public void changeTheColor()
    {
        if(changeColor)
        {
           for(int i = 0; i < 3; i ++)
           {
                 if(rgb[i] < 255)
                 {
                       rgb[i] += 1;
                 }
                 else
                 {
                       rgb[i] = 0;
                 }

           }
        }
    }
}
```

GUI

Message Dialog

GUI is another way to output the programs, and input.
Use the SWING library

```
1       import javax.swing.JOptionPane;
2
3       public class RunDialogs {
4
5       public static void main(String[] args) {
6
7       JOptionPane.showMessageDialog(null, "Welcome to
8       Java!");
9
10      }
11
        }
```

```
//-----------------------------------------------------------
import javax.swing.JOptionPane;
public class MessageDialogue {

    public static void main(String[] args) {
       // TODO Auto-generated method stub

       JOptionPane.showMessageDialog(null, "Welcome to Java!", "Welcome",
JOptionPane.WARNING_MESSAGE);

    }

}
```

showMessageDialog(parent window(null), message (string), window itle(string), type of icon to add the dialog)

Types of messages:

- JOptionPane.INFORMATION_MESSAGE
- JOptionPane.WARNING_MESSAGE
- JOptionPane.ERROR_MESSAGE

Graphic User Interface

JFrame is a java class that lets you set window size and other properties

```
1     import javax.swing.JFrame; //imports JFrame
2
3     public class MyWindow extends JFrame{ //extends j frame
4
5     //constructor
6     MyWindow(String title){
7
8     super(title); //calls the constructor of the parent class JFrame
9
10    this.setSize(400, 650);
11    this.setVisible(true);
12    }//end constructor
13    }
```

```
1     import javax.swing.JFrame;
2
3     public class JavaWindow {
4
5         public static void main(String[] args) {
6            // TODO Auto-generated method stub
7
8            MyWindow window = new MyWindow("Hello");
9
10              // Set the program to close when the window is
11    closed
12
13    window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
14        }

      }
```

To create a new object you need to create a new instance of the previous program

Labels and Text Fields

Object Reference:

```
1     //import label and text fields
2     import javax.swing.JFrame;
3     import javax.swing.JLabel;
4     import javax.swing.JPanel;
5     import javax.swing.JTextField;
6
7     public class MyWindow extends JFrame {
8       JTextField name, strength, defense, health; //these
9     are the input tags
10
11        //constructor
12        MyWindow(String title){
13
14           super(title);
15
             this.setSize(400, 650);
             this.init();
             this.setVisible(true);
           }//end constructor
         void init(){ //initialize tags in here

           JLabel lname = new JLabel("Name:"); //creates
           new label with name
            JLabel lstrength = new JLabel("Strength:");
           JLabel ldefense = new JLabel("Defense:");
           JLabel lhealth = new JLabel("Health:");


           name = new JTextField(10);
           //creates text field 10 chars long
           strength= new JTextField(10);
           defense= new JTextField(10);
           health = new JTextField(10);


           //create a panel, add the label for name, the
           input for the name, and the actual panel
           JPanel panel = new JPanel();
           panel.add(lname);
           panel.add(name);
```

```
        panel.add(lstrength);
        panel.add(strength);
        panel.add(ldefense);
        panel.add(defense);
        panel.add(lhealth);
        panel.add(health);
        this.add(panel);

    }


    }
```

```
1   import javax.swing.JFrame;
2
3   public class JavaWindow {
4
5       public static void main(String[] args) {
6          // TODO Auto-generated method stub
7
8          MyWindow window = new MyWindow("Hello");
9
10  window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
11      }
12
    }
```

Buttons and Actions

Buttons cause actions to happen. to import a button use JButton. Then listen to see if the button is clicked by using actionlistener, action event.

you have to exten action listener in the class

```
import java.swing.JButton
```