As you arrive, please snarf the code for this recitation.

Then discuss this:

```java
public class HangmanExecuter {
    public static void main(String[] args) {
        HangmanGame game = new HangmanGame();
        game.play();
    }
}
```

In the code above, why can’t we replace the two lines of code with `HangmanGame.play();`?

A. The code would not compile.
B. It's bad style – because Java is an Object-Oriented language we really should be creating objects and not just calling functions
C. We might someday want to have several instances of `HangmanGame` at once, each with their own fields
D. It would be fine to replace those lines

As you arrive, please snarf the code for this recitation.

Then discuss this:

```java
public class PreAssignment {
    private int myField;
    public PreAssignment(String parameter)
    {
        System.out.println("I am in a constructor and my parameter is: "+ parameter);
    }
    public void method()
    {
        System.out.println("I am in a method!");
    }
    public static void main(String[] args) {
        PreAssignment p = new PreAssignment("Hello");
        p.method();
    }
}
```

What You Will Be Able to Do After Class

- Explain why we need constructors in the first place
  1. I’ll ask for questions about the prep-assignment
  2. We will discuss the answer to the question I posted at the beginning of class (several answers will turn out true)
  3. I’ll talk (very briefly) about the point of all these objects and how constructors fit into that
- Avoid a common bug having to do with mutable objects
  1. There will be trivy multiple choice questions to answer!
- Fix bugs with your APTs
  1. On the Server
  2. And Locally

My Solution to the Prep Assignment

```java
public class PreAssignment {
    private int myField;
    public PreAssignment(String parameter)
    {
        System.out.println("I am in a constructor and my parameter is: "+ parameter);
    }
    public void method()
    {
        System.out.println("I am in a method!");
    }
    public static void main(String[] args) {
        PreAssignment p = new PreAssignment("Hello");
        p.method();
    }
}
```
What’s the deal with classes?

1. Organize code in a way that’s easy to understand
2. “Encapsulates” data together with functions in a way that is difficult with functions alone
3. Allows certain kinds of abstract structures to be built easily (not appearing in this lecture)

Pop Quiz!

Is there a method in Java that can find the last occurrence of a particular String in a list of Strings?

“Encapsulating” Stuff

Functions “encapsulate” how to do particular things:

public boolean isPrime(int num) {
    // some pretty heavy math is involved in computing this
}

Problem: Sometime Data is Complicated Too

/* This function takes an array of points representing a 3D object, and magnitudes w, x, y, z representing a rotation quaternion and determines the result of the 3D rotation */
public int[][][] rotate(double w, double x, double y, double z, int[][][] input)

public Shape3D rotate(Rotation3D rot, 3DObject objToRotate)
Objects “Encapsulate” data together with functions in a way that is difficult with functions alone. It lets programs hide details of how it keeps data.

But it’s common for data to require some kind of initialization.

(Live hangman code modification demo goes here)

Where we are

- Explain why we need constructors in the first place

**Why do we need constructors? (write it on your sheets)**

**Coming Next:**
- How to avoid a common bug having to do with mutable objects
  1. There will be tricky multiple choice questions to answer!
- How to fix bugs with your APTs
  1. On the Server
  2. And Locally

```java
public class Rec2Example {
    private int myA;
    private int myB;
    public Rec2Example(int a, int b) {
        myA = a;
        myB = b;
    }
    public static void main(String[] args) {
        Rec2Example ex = new Rec2Example(33,7);
        Rec2Example ex2 = ex;
    }
}
```

How many times is the Rec2Example constructor called in main?

A. 0 Times
B. 1 Times
C. 2 Times
D. 3 Times

```java
public class Rec2Example {
    private int myA;
    private int myB;
    public Rec2Example(int a, int b) {
        myA = a;
        myB = b;
    }
    public void setA(int a) {
        myA = a;
    }
    public static void main(String[] args) {
        Rec2Example ex = new Rec2Example(33,7);
        Rec2Example ex2 = ex;
        ex.setA(44);
    }
}
```

What is the value of ex2.myA at the end of main?

A. 33
B. 44
C. 7
D. The greatest common factor of 33, 4, and 7
public static void randomFunction() {
    ArrayList<Integer> list = new ArrayList<Integer>();
    ArrayList<Integer> otherList = list;
    list.add(44);
    otherList.add(55);
}

What does otherList contain at the end of randomFunction?
A. [44]  
B. [55]  
C. [44, 55]  
D. None of these; the function won't compile because it is static

public static void randomFunction2() {
    String a = "Hello";
    String b = "Goodbye";
    b = a;
    a.concat(" CS100");
}

What is true at the end of randomFunction2?
A. b contains "Hello CS100"  
B. b contains "Goodbye" because strings are immutable  
C. b contains "Hello" and a will contain "Hello CS100" because string are immutable  
D. Both b and a contain "Hello" because strings are immutable

Where we are

• You can hopefully explain that we need constructors to ensure that objects are initialized properly
• You hopefully now will not have bugs from accidentally modifying mutable objects
• Coming Next: How to fix bugs with your APTs
  1. On the Server
  2. And Locally