Java Review

The stuff you should already know.
What I Expect

- You should already know programming basics.

- If you don’t know Java, that’s ok, but you must study it!
Installing and Running Java

- Read my online notes from “Introduction to Programming”
  - [http://academic.regis.edu/dbahr/](http://academic.regis.edu/dbahr/)

- Follow directions to
  - Install Java.
  - Test Java by running and compiling.
  - Install Eclipse if desired.
public class BasicProgram
{
    public static void main(String[] args)
    {
        System.out.println("Yo, dude.");
    }
}
For Loop

```java
for(int i=0; i<200; i++)
{
    System.out.println("i = "+i);
}

//go back in time
for(int age=100; age >=0; age--)
{
    System.out.println("age = "+age);
}
```

Comment
boolean happy = false;
while(!happy) {
    String sAmount = JOptionPane.showInputDialog("Enter amount of snowfall.");
}

Must “import javax.swing.JOptionPane;”

How many times does this run?
boolean happy = false;
while(!happy)
{
    String sAmount = JOptionPane.showInputDialog("Enter amount of snowfall");
    //convert to a double
    double dAmount = Double.parseDouble(sAmount);
    if(dAmount < 12.0)
    {
        happy = true;
    }
}
**Boolean Expressions**

- *if* and *while* can take any boolean expression.
  - `true`
  - `false`
  - `evil` (where “boolean evil = true”)
  - `i < 10`
  - `i < 10 && i > 4`
    - what integers can i be to get a “true”?
  - `i < 10 || j > 20`
  - `i == 5`
    - how is this different from “=”?
    - Java does not accept “i = 5” as a boolean expression
boolean ateWormForLunch = true;
if(ateWormForLunch == true)
{
    ...
}

boolean ateWormForLunch = true;
if(ateWormForLunch)
{
    ...
}

Both work. This one is clearer!
Methods

- Break up different tasks

```java
public class Dog {
    public String bark() {
        return "Bow-wow!";
    }

    public double calculateSqrt(double dNumber) {
        return Math.sqrt(dNumber);
    }
}
```

What do you think?
Should this be in a Dog object?
public class SmartDogDemo
{
    public static void main(String[] args)
    {
        Dog rover = new Dog();
        System.out.println( rover.bark() );
        double squareRoot = rover.calculateSqrt(25.0);
    }
}
Methods in Main Class

```java
public class Yak {
    public static void main(String[] args) {
        Yak suzy = new Yak();
        System.out.println( suzy.grunt() );
    }

    public String grunt() {
        return "unghhh";
    }
}
```

- Instantiates this class!
- Called bootstrapping.
- Now can use methods in this class.
Suppose Dog “grunts” just like a Yak.

```java
public class Dog {
    public String bark() {
        return "Bow-wow!";
    }

    public String grunt() {
        Yak tim = new Yak();
        String sNoise = tim.grunt();
        return sNoise;
    }
}
```
Local Variables

- Any variable declared inside a method can’t be used elsewhere.

```java
public class Yak {
    public static void main(String[] args) {
        System.out.println(noise);
    }
    public String grunt() {
        String noise = "unghhh";
        return noise;
    }
}
```
public class Example
{
    public static void main(String[] args)
    {
        double dValue = 23.1;

        Example e = new Example();
        e.change(dValue);

        System.out.println(dValue);
        System.out.println(dNumber);
    }

    public void change(double dNumber)
    {
        dNumber = 999.99;
    }
}
import javax.swing.JOptionPane;
public class ArrayExample
{
    public static void main(String[] args)
    {
        double[] dArray = new double[10];
        for(int index = 0; index < 10; index++)
        {
            dArray[index] = 0.0;
        }
        dArray[3] = 345.67;
        JOptionPane.showMessageDialog(null, "dArray[3] = " + dArray[3]);
    }
}
public static void main(String[] args) {
    Dog[] dogPack = new Dog[47];
    for(int alphaIndex = 0; alphaIndex < 47; alphaIndex++) {
        dogPack[alphaIndex] = new Dog();
    }
    System.out.println( dogPack[3].bark() );
    System.out.println( dogPack[21].bark() );
}

Declaration of array. Just reserves the space.

Instantiation of each Dog in the array. Very important! Tells it what to put in the space.

Can’t use the array, until the Dog at position 3 is declared.

Note the use of the Dog’s method.
Returning Arrays From Methods

public char[] jacobLetters() {
    char[] jacob = {"j", "a", "c", "o", "b"};
    return jacob;
}

return type is an array of char

Note implicit sizing of array. Will have 5 elements.

must return an array of char
Pointers

A pointer is just a reference to a thing.
- e.g., Instead of holding the tree, I’m just pointing at the tree.
- Actually a variable that holds a memory address.
  - The tree would be stored at that memory address.

All complex variables are pointers.
- i.e., All instantiated classes are pointers.
- e.g., Dog spot = new Dog();
  - The variable “spot” is really just a pointer to the dog.
  - The actual Dog is at the memory address held by “spot”
Arrays Are Pointers

- `char[] cArray = new char[15];`

- `cArray` holds the memory address of where the 15 characters will be stored.
No longer local. (gasp!)

```java
public class PointerExample {
    public static void main(String[] args) {
        char[] name = {'g', 'a', 'r', 'y'};

        PointerExample pe = new PointerExample();
        pe.changeCharacters(name);
        for (int i = 0; i < name.length; i++) {
            System.out.print(name[i]);
        }
    }

    public void changeCharacters(char[] cArray) {
        cArray[1] = 'o';
    }
}
```

What gets printed?

What’s this do?
Methods may be declared public, private, protected, or nothing at all.

- **public** means that any other class can access it.
- **private** means that only its own class can access it.
- **protected** means only child classes can use it
  - We’ll explore this concept of “children” much later.

- nothing – means that any program in the same folder can see it.
  - Java calls folders by the fancy name “packages”
  - A dangerous privilege!
Public, Private Methods

```java
public class Yak {
    private int randomNumber() {
        Random r = new Random();
        return r.nextInt(255);
    }

    public String grunt() {
        return "unghhh";
    }

    public String randomNoise() {
        return "" + (char) randomNumber();
    }
}
```

No other class can use this method.

Any other class can use these methods.

But Yak can use its own private method!

Whoa! What’s this do? (casting to ascii...)
Considering the previous slide, which are valid method calls?

```java
public static void main(String[] args) {
    Yak yikkity = new Yak();

    yikkity.grunt();
    yikkity.randomNoise();
    yikkity.randomNumber();
}
```