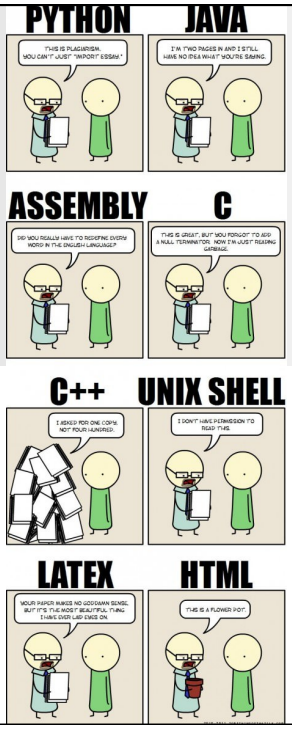


An intro to programming*

*Java

Announcements

- Professor Rodger
- The Link
  - UTAs start TONIGHT!

2

Announcements

- Office hours –
  - Check the website!

3

Homework

- Recitation prep due BEFORE recitation on Friday
  - Setup and start coding in Java
    - Post questions on Piazza
- APT Set 1 (Java coding)
  - Due September 3
- Assignment 1 – (More Java coding)
  - Due September 9
  - More discussion on Monday

4

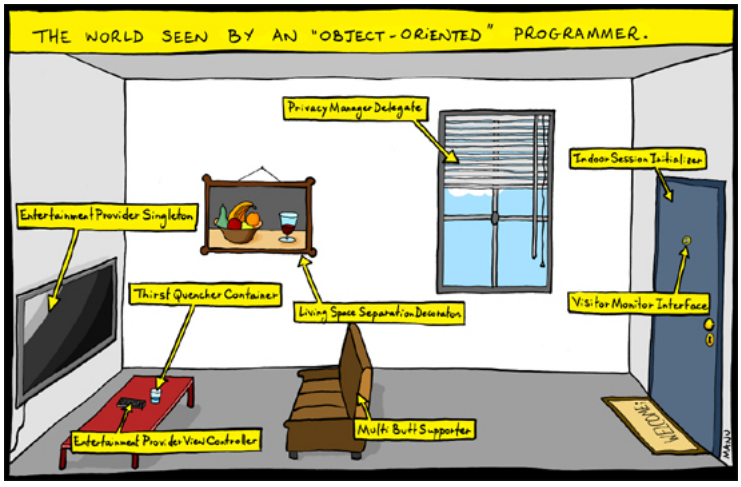
Today

- JAVA JAVA JAVA JAVA JAVA JAVA
  - JAVA
    - Primitive and Object type
    - Classes
    - Code example
  - Java
    - Arrays

5

Java

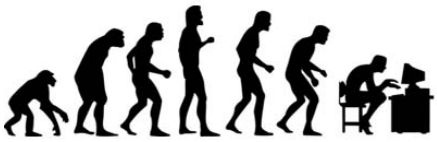
- Object-Oriented Programming Language



6

Primitive types

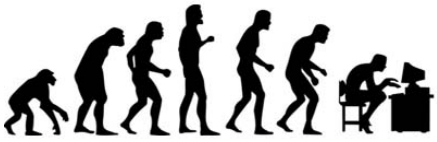
- int
  - `int myInt = 5;`
- char
  - `char myChar = 't';`
- boolean
  - `boolean myBool = false; // or true`
- double
  - `double myDouble = 5.5;`

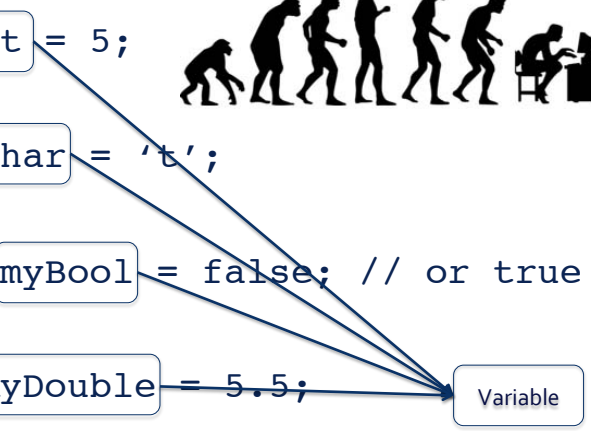


7

Primitive types

- int
  - `int myInt = 5;`
- char
  - `char myChar = 't';`
- boolean
  - `boolean myBool = false; // or true`
- double
  - `double myDouble = 5.5;`

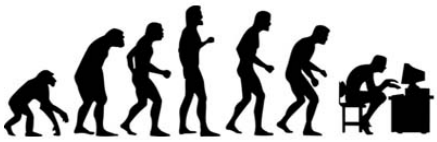




8

Primitive types

- int
  - `int myInt = 5;`
- char
  - `char myChar = 't';`
- boolean
  - `boolean myBool = false; // or true`
- double
  - `double myDouble = 5.5;`



Type

9

Primitive types

- value of primitive stored in memory
- know memory size

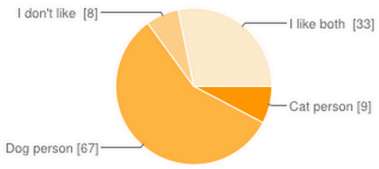
```
int myInt = 5;
char myChar = 't';
boolean myBool = false;
double myDouble = 5.5;
```

myInt	5
myChar	't'
myBool	false
myDouble	5.5

10

Cats and Dogs

Are you a cat person or a dog person?



Cat person	**9**	8%
Dog person	**67**	57%
I don't like animals	**8**	7%
I like both cats and dogs equally	**33**	28%

11

Object type

- Object – has *state* and *behavior*
  - Dog
    - State – name, breed, age
    - Behavior – barking, eating, sleeping

12

Code

```
1  public static void main(String[] args){
2        Dog max = new Dog("Mad Max", 7, "mutt");
3
4        max.sleeping();
5
6     }
```

Code

```
1  public static void main(String[] args){
2        Dog max = new Dog("Mad Max", 7, "mutt");
3
4        max.sleeping();
5
6     }
```

State

Behavior

Class

```
1   public class Dog {
2       String myName;
3       int myAge;
4       String myBreed;
5
6       void barking(){
7           System.out.println(myName + "is barking. Woof Woof");
8       }
9
10      void eating(){
11          System.out.println(myName + "is eating. Yum yum");
12      }
13
14      void sleeping(){
15          System.out.println("shhh " + myName + " is sleeping.");
16      }
17  }
```

Blueprint for an object

15

Class

```
1   public class Dog {
2       String myName;
3       int myAge;
4       String myBreed;
5
6       void barking(){
7           System.out.println(myName + "is barking. Woof Woof");
8       }
9
10      void eating(){
11          System.out.println(myName + "is eating. Yum yum");
12      }
13
14      void sleeping(){
15          System.out.println("shhh " + myName + " is sleeping.");
16      }
17  }
```

State (Instance variables)

16

Class

```
1   public class Dog {
2       String myName;
3       int myAge;
4       String myBreed;
5
6       void barking(){
7           System.out.println(myName + "is barking. Woof Woof");
8       }
9
10      void eating(){
11          System.out.println(myName + "is eating. Yum yum");
12      }
13
14      void sleeping(){
15          System.out.println("shhh " + myName + " is sleeping.");
16      }
17  }
```

State (Instance variables)

Behavior (Methods)

17

Code

```
1   public static void main(String[] args){
2           Dog max = new Dog("Mad Max", 7, "mutt");
3
4           max.sleeping();
5
6       }
```

18

Class

```
1  public class Dog {
2      String myName;
3      int    myAge;
4      String myBreed;
5
6      public Dog(String name, int age, String breed){
7          myName = name;
8          myAge = age;
9          myBreed = breed;
10     }
11
12     void barking(){
13         System.out.println(myName + "is barking. Woof Woof");
14     }
```

State (Instance variables)

Behavior (Methods)

19

Class

```
1  public class Dog {
2      String myName;
3      int    myAge;
4      String myBreed;
5
6      public Dog(String name, int age, String breed){
7          myName = name;
8          myAge = age;
9          myBreed = breed;
10     }
11
12     void barking(){
13         System.out.println(myName + "is barking. Woof Woof");
14     }
```

State (Instance variables)

Create an Object (Constructor)

Behavior (Methods)

20

Code

```
1  public static void main(String[] args){
2        Dog max = new Dog("Mad Max", 7, "mutt");
3
4        max.sleeping();
5
6     }


public Dog(String name, int age, String breed){
    myName = name;
    myAge = age;
    myBreed = breed;
}
```

Create an Object (Constructor)

21

Code

```
1  public static void main(String[] args){
2        Dog max = new Dog("Mad Max", 7, "mutt");
3
4        max.sleeping();
5
6     }


public Dog(String name, int age, String breed){
    myName = name;
    myAge = age;
    myBreed = breed;
}
```

Create an Object (Constructor)

22

Code

```
1   public static void main(String[] args){
2         Dog max = new Dog("Mad Max", 7, "mutt");
3
4         max.sleeping();
5
6       }

void sleeping(){
       System.out.println("shhh " + myName + " is sleeping.");
   }
```

shhh Mad Max is sleeping.

23

APT demo

- Gravity
  - http://www.cs.duke.edu/csed/newapt/gravity.html

APT: Gravity

Problem Statement

Elphaba has decided to try to defy gravity. She's going to drop or throw an object from the top of an infinitely tall building and see how far it falls. She knows exactly what speed she throws the object and has a stop watch she uses to time how long it falls.

Write method `falling` that returns the number of *meters* the object has fallen after `time` seconds have elapsed when the object is thrown with an initial velocity of `velo` meters/sec.

Ignore any forces due to friction, air-resistance, etc. The infinitely tall building is located in a vacuum. However, the building is on the earth, so accelaration due to earth's gravity should be part of your calculations.

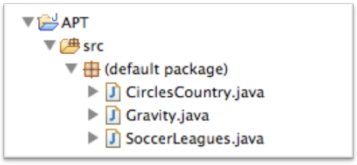
Class

```
public class Gravity {
    public double falling(double time, double velo){
      // fill in code here
    }
}
```

24

APTs

- Make APT java project
  - Add new class for each APT
  - Class / method names EXACTLY match assignment
    - Helper methods can be named anything
  - Test online many times



- Testing/Debugging
  - Output appears in online APT tester
  - Write your own `main`
  - Use the debugger!

25

APT Grading

- Complete required number by due date
- Not keeping up with APTs hurts final APT grade
  - 1/2 of APT grade is completing APTs on time
- "Required" is fair game for exams / recitation
- If you skip an APT you can go back!
  - Keep trying. Hand it in later!

26

APT Grading

- You can always do more than is required!
  - This will help your grade!
- If you fall behind once - make up for it next time.
  - This won't hurt your grade. (And please don't ask me about it)
- Handing in all APTs at the end of the semester WILL hurt your grade!
  - And could hurt your test grades.

27

Arrays

- An Array – a collection of items selected by index
  - fixed number of values
  - single type

```
char[] duke = new char[5];
duke[0] = 'd';
```

0	1	2	3	4
'd'	'e'	'v'	'i'	'l'

28

Arrays

- How do you create an array of 10 Strings?

```
a. int[] a = new String[10];
b. String[] b = new String[9];
c. String[10] c = new String[10];
d. String[] d = new String[10];
```

29

Arrays

- Fixed length
  - `int[] a = new int[5];`
  - 5 is the length
- type specific
  - int, char, String, double, Dog, etc.
- Access length of an array
  - `int len = a.length;`

Can you have an array with a varying length?

Is there a tradeoff?

*Note: arrays are Objects - you can make function calls

30

Arrays

- Coding!
  - Create class `ArrayPractive.java`
  - Add method `makeArray` that
    - Creates an array of 50 doubles
    - Puts the number 20.5 into every entry
    - Returns the array

31

Recitation

- Recitation assignment due BEFORE recitation on Friday
  - Setup and start coding in Java
  - Read CirclesCountry and THINK about how to solve it

32