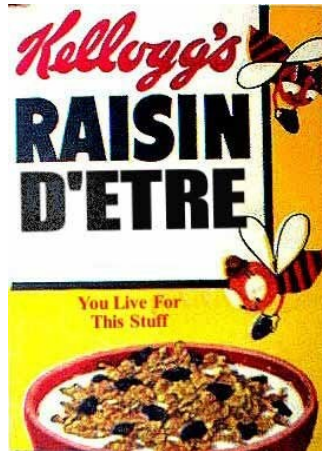
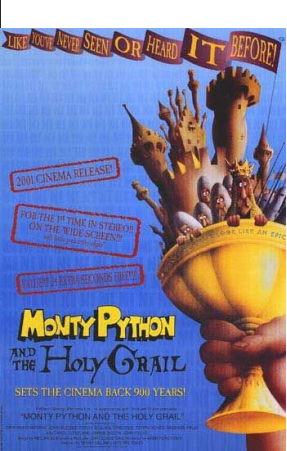


Java and Object-Oriented Programming



Declaring the BA CS Major?

- Want to declare the BACS major before advising and course selection start in a few weeks?
- There will be an Infosession this Thursday from 5-6pm in Rice 340
 - Just show up!
 - Or email horton@cs.virginia.edu for details.

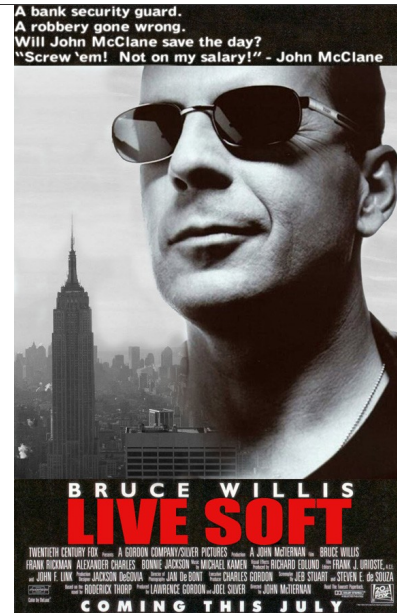
One-Slide Summary

- Real databases, unlike PS5, have many concerns, such as scalability and atomic transactions.
- A **type** is a (possibly infinite) set of values.
- Each type supports a set of **valid operations**.
- Types can be latent or manifest, static or dynamic, strong or weak.
- An **object** packages state and procedures.
- A procedure on an object is called a **method**. We **invoke** a method by sending the object a **message**.
- **Inheritance** allows one object to refine and reuse the behavior of another. This is a good thing.
- Java is **statically-typed** and **object-oriented**.

#3

Outline

- PS5 vs. the Real World
- Problem Sets and PS9
- Types
- Java
- Object-Oriented Programming
 - Object = State + Methods
- Inheritance



Interlude: PS5 vs. Wild

How are commercial databases different from what you implemented for PS5?

UVA's Integrated Systems Project to convert all University information systems to use an Oracle database was originally budgeted for **\$58.2 Million** (starting in 1999). Actual cost ended up over \$100 Million.

<http://www.virginia.edu/isp/>

#5

Real Databases

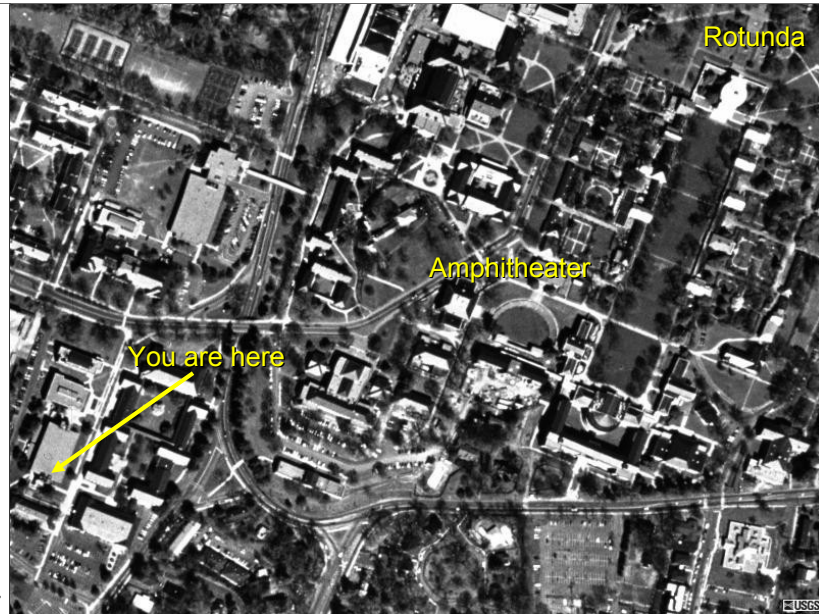
- **Atomic Transactions:** a transaction may involve many modifications to database tables, but the changes should only happen if the whole transaction happens (e.g., don't charge the credit card unless the order is sent to the shipping dept)
- **Security:** limit read/write access to tables, entries and fields
- **Storage:** need to efficiently store data on disk, provide backup mechanisms
- **Scale:** to support really big data tables, real databases do lots of clever things

#6

How big are big databases?

- Microsoft TerraServer

- Claimed biggest in 1998
- Aerial photos of entire US (1 meter resolution)
- Let's see an example ...



#7

Big Databases

- Microsoft TerraServer

- 3.3 Terabytes (claimed biggest in 1998)
- 1 Terabyte = 2^{40} Bytes ~ 1 Trillion Bytes

- Google Maps (possibly bigger?)

- Better color ...

- Wal-Mart

- 285 Terabytes (2003)



- Stanford Linear Accelerator (BaBar)

- 500 Terabytes (30 KB per particle collision)

#9

How much work?

- Suppose we have a huge database.
- table-select is in $\Theta(n)$ where n is the number of entries in the table
 - Would your table-select work for Wal-Mart?
 - If 1M entry table takes 1s, how long would it take Wal-Mart to select from 285TB ~ 2 Trillion Entries?

#10

How much work?

- table-select is in $\Theta(n)$ where n is the number of entries in the table
 - Would your table-select work for Wal-Mart?
 - If 1M entry table takes 1s, how long would it take Wal-Mart to select from 285TB ~ 2 Trillion Entries? 2 000 000s = ~ 23 days

How do expensive databases perform table-select so much faster?

Hint: How did we make sorting faster?

#11

Objects

An **object** packages:

- **state** ("variables")
- **procedures** for manipulating and observing that state ("methods")

Why is this useful?

#12

Problem-Solving Strategies

- PS1-PS4: **Functional Programming**
 - Focused on **procedures**
 - Break a problem into procedures that can be combined to solve it
 - All Python
- PS5: **Imperative Programming**
 - Focused on **data**
 - Design data for representing a problem and procedures for updating that data
 - All Python + Small Java Intro

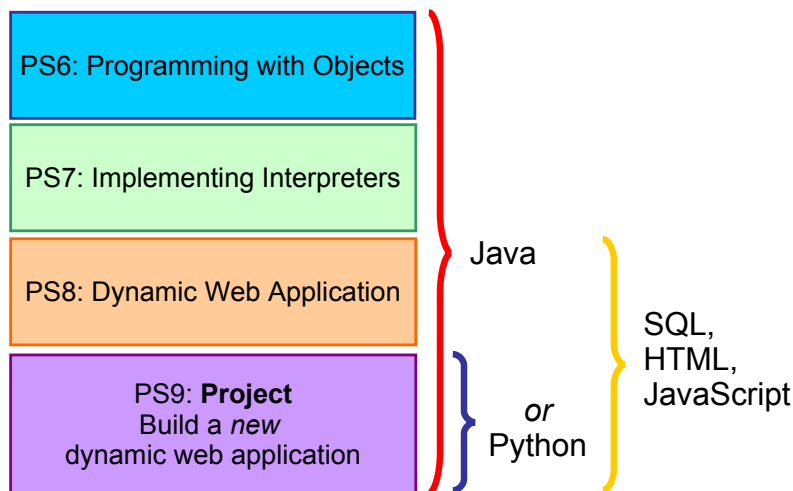
#13

Problem-Solving Strategies

- PS6: **Object-Oriented Programming**
 - Focused on **objects**: package procedures and state
 - Model a problem by dividing it into objects
 - Lots of problems in real (and imaginary) worlds can be thought of this way
 - All Java

#14

Problem Sets after PS5



#15

PS9 Assignment

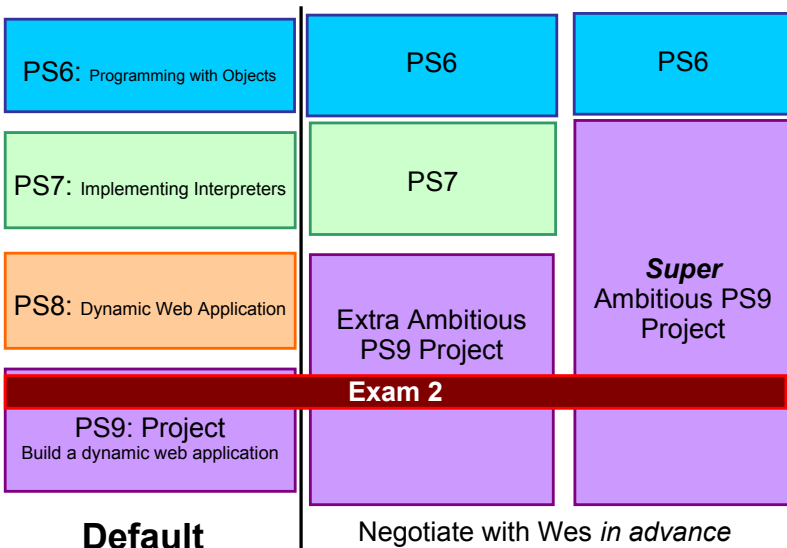
Problem: Make an interesting dynamic web site.

- Teams of 1-50 students
- Can be anything you want that:
 - Involves interesting computation
 - Follows University's use policies (or on external server)
 - Complies with ADA Section 508 (accessible)



A list of example topics is provided.

#16



#17

Liberal Arts Trivia: Biology

- This egg-laying, venomous (from a calcaneus spur found on the hind limb), beaver-tailed, otter-footed mammal is perhaps best known for its “nose”, which follows the style of the Anatidae family of birds. It is native to eastern Australia and Tasmania, and occurs on the Australian 20 cent coin.

#18

Liberal Arts Trivia: Art History

- Name the Spanish surrealist artist who painted *The Persistence of Memory* (oil on canvas, 1931).



Most Popular Programming Languages

1. Java
2. C
3. PHP
4. C++
5. Visual Basic
6. C#
7. Python
8. Perl
9. Delphi
10. JavaScript

TIOBE
Index, March 2010

#20

The Reveal

- Java is almost identical to Python
 - Both have variables, if-else, function definitions, recursion, `mylist[3] = 44`, ways to print things, etc.
- Syntactic Differences:
 - Python is **concise** and uses `:` and `[Tab]`
 - Java is **verbose** and uses `;` and `{ }`
- Semantic Differences:
 - Java uses **Types** to notice errors.
 - Java uses **Objects** to organize state and functions.

#21

Latent Python Danger

```
def mydouble(x):  
    if get_date() != 'Saturday':  
        return x * 2  
    else:  
        return x * "two"  
  
print mydouble(3)  
# What will you get if you run it today?  
# Are there any bugs in this program?
```

#22

Types



#23

Types

Integers

Strings

programs that halt

Colors

Beatle's Songs that don't end on the Tonic

lists of lists of lists of Strings

- A **Type** is a (possibly infinite) set of values
- You can do some things with some types, but not others
 - Each Type has associated **valid operations**

#24

Why have types?

- Detecting programming errors: (usually) better to notice error than report incorrect result
- Make programs easier to read, understand and maintain: thinking about types can help understand code
- Verification: types make it easier to prove properties about programs
- Security: can use types to constrain the behavior of programs

#25

Types of Types

Does regular Python have types?

```
>>> 3[0]
TypeError: 'int' object is not subscriptable
>>> "hello" + 2
TypeError: cannot concatenate 'str' and 'int' objects
```

Yes, without types `3[0]` would produce some silly result. Because of types, it produces a type error.

#26

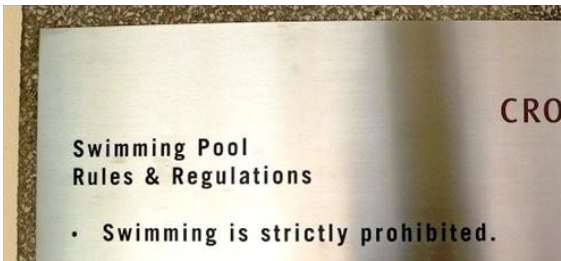
Python Sees Types When Running

```
>>> 3 + "hello"
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unsupported operand type(s) for +: 'int' and 'str'



#28

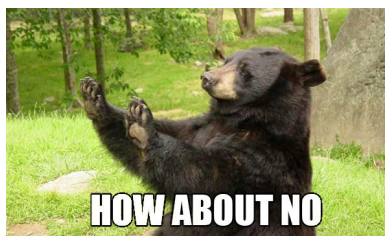
Type Taxonomy

- **Latent** vs. **Manifest**
 - Are types visible in the program text?
- **Static** vs. **dynamic** checking
 - Do you have to run the program to know if it has type errors?
- **Weak** vs. **Strong** checking
 - How strict are the rules for using types?
 - (e.g., does the predicate for an if need to be a Boolean?)
 - Continuum (just matter of degree)

#28

Scheme/Python/Charme

- Latent or Manifest?
 - All have **latent** types (none visible in code)
- Static or Dynamic?
 - All are **dynamic** (checked when expression is evaluated)
- Weak or Strong?
 - Which is the strictest?
 - You tell me!



Strict Typing

```
Java> 1 + (5 > 3)
```

operator + cannot be applied to int,boolean

```
Python>>> 1 + (5 > 3)
```

```
2
```

```
Scheme> (+ 1 (> 5 3))
```

```
2
```

```
C> 1 + (5 > 3)
```

```
2
```



Python → Java

- Python (and Scheme) have Latent, Dynamically checked types
 - Don't see explicit types when you look at code
 - Checked when an expression is evaluated
- Java has **Manifest, Statically checked** types
 - Type declarations must be included in code
 - Types are checked statically before running the program (Java: not all types checked statically)

#31

Python vs Java


```
def sum(x):
    t = 0 # running total
    for i in range(len(x)):
        t = t + x[i]
    return t

lst = [1,2,3];
print sum(lst);
```

```
public class PS5 {
    public static int sum(int [] x) {
        int t; // running total
        for (int i=0; i<x.length; i=i+1) {
            t = t + x[i];
        }
        return t;
    }
    public static void main(Strings [] args) {
        int [] lst = {1, 2, 3};
        System.out.println(sum(lst));
    }
}
```

#32

Java Example 2



```
class Test {
    int tester (String s)
    {
        int x;
        x = s;
        return "okay";
    }
}
```

The result is an integer → `int tester`

The parameter must be a String → `(String s)`

The place x holds an integer → `int x;`

#33

Java Example

```
> javac types.java
types.java:5: Incompatible
type for =. Can't convert
java.lang.String to int.
    x = s;
    ^
types.java:6: Incompatible
type for return. Can't convert
java.lang.String to int.
    return "okay";
    ^
2 errors
```

```
class Test {
    int tester (String s)
    {
        int x;
        x = s;
        return "okay";
    }
}
```

The result is an integer → `int tester`

The parameter must be a String → `(String s)`

The place x holds an integer → `int x;`

`javac compiles (and type checks) the program. It does not execute it.`

#34

Why Learn New Languages?

- Languages change the way we think.
 - The **linguistic relativity principle** (also known as the **Sapir-Whorf Hypothesis**) is the idea that the varying cultural concepts and categories inherent in different languages affect the cognitive classification of the experienced world in such a way that speakers of different languages think and behave differently because of it. Roger Brown has drawn a distinction between weak linguistic relativity, where **language limits thought**, and strong linguistic relativity, where language determines thought. [Wikipedia]
- See also: Orwell's *1984*

#35

Why Learn New Languages?

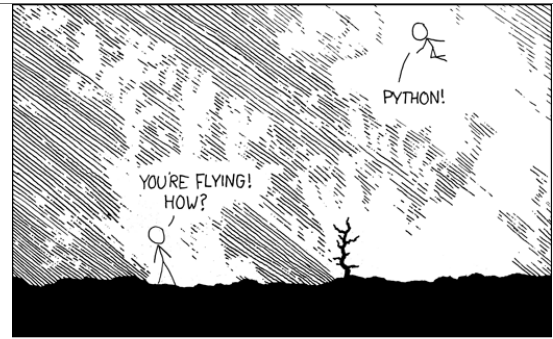
- Deepening Understanding
 - By seeing how the same concepts we encountered in Scheme are implemented by a different language, you will **understand those concepts better** (especially procedures, assignment, and data abstraction).
- Building Confidence
 - By learning Java (mostly) on your own, the next time you encounter a problem that is best solved using a language you don't know, you will **be confident you can learn it** (rather than trying to use the wrong tool to solve the problem.)

#36

Why Learn New Languages

- Fun! Programming in can be Java is fun.
- Especially because:
 - It is commonly-used to solve real-world problems.
 - It is well-suited to group work.
 - It makes it easy to catch errors in advance.
 - It is strongly object-oriented.
 - They were going to name it “Oak” after the tree outside the office window, but that was already trademarked.

#37



#38

Java

- **Java** is a **universal programming language**.
 - Everything you can compute in Python you can compute in Java, and vice versa
 - PS 7: implement a Python interpreter in Java
 - Chapter 12: more formal definition of a universal programming language
- Java is an **imperative language**.
 - Designed to support programming where most of the work is done using **assignment statements**
 - `x = sqrt(4) + 1;`

#39

Objectifying Java

- Java is also an **object-oriented language**.
 - **Objects** encapsulate **state** (i.e., variables and information) and the **methods** that operate on that state together.
 - In Java, almost all data are objects.
 - Problem Set 6 covers programming with objects.
 - Java has built-in support for classes, methods and inheritance.

#40

Learning New Languages

- **Syntax**: Where the {, !, (, :, etc., all go
 - If you can understand a BNF grammar, this is easy
 - But it still takes some getting used to
- **Semantics**: What does it mean?
 - Learning the evaluation rules
 - This is harder, but most programming languages have very similar rules (with subtle differences)
- **Style**: What are the idioms and customs?
 - Many years to be a “professional” Java programmer, but not long to write a program

#41

Java If

- **Instruction ::=**
if (Expression) {
 Block
} else {
 Block
}
- **Semantics**: Evaluate the *Expression* (which must be a Boolean). If it evaluates to a true value, evaluate the first *Block*. Otherwise, evaluate the second *Block*.
- You can omit **else { ... }**

#42

Java If Example

```
if (this_one > best_sofar) {
    System.out.println("This one is better!");
} else {
    System.out.println("Not better!");
}
```



Learning Java

- We will introduce (usually informally) Java constructs in class as we use them (and in example code in PS5 and PS6)
- The “Java Lab Guide” is a video introduction to Java and Eclipse:
 - Java : Eclipse :: Python : PyCharm
 - Covers what you need for PS5.
- On-line Java documentation

#44

Making Objects

```
ClassDefinition ::=
public class Name {
    FunctionOrFieldDefinitions
}
```

```
public class Dog {
    public static void bark() {
        System.out.println("wuff wuff wuff");
    }
}
```

*In Washington, it's dog eat dog.
In academia, it's exactly the opposite.*
- Robert Reich

#45

Making a Dog

```
Expression ::= new ClassName(args)
```

```
public class Dog {
    public static void bark() {
        System.out.println("wuff wuff wuff");
    } }
...
Dog spot = new Dog();
```

In Java, you must *declare* a variable with its type before you give it a value.

#46

Java Procedures (= Methods)

```
public class Dog {
    public static void bark() {
        System.out.println("wuff wuff wuff");
    } }
}
```

```
MethodDefinition ::= Modifiers Type Name ( Params ) Block
Params ::= SomeParams | epsilon
SomeParams ::= Type Name | Type Name, SomeParams
Block ::= { Statements }
Statements ::= Statement ; MoreStatements
MoreStatements ::= epsilon
                 | Statement ; MoreStatements
```

#47

Some Java Procedures

```
int square(int x) {
    return x*x;
}

int bigger(int a, int b) {
    if (a > b)
        return a;
    else return b;
}

int biggest(int [] lst) {
    int biggest = lst[0];
    for (int i = 1; i < lst.length; i = i + 1)
        if (lst[i] > best) biggest = lst[i];
    return biggest;
}
```

#48

Barking: Invoking Methods

```
ApplicationStmt ::= Expr.Name(Args)
Args ::= epsilon | MoreArgs
MoreArgs ::= Argument , MoreArgs
           | Argument
Argument ::= Expr
```

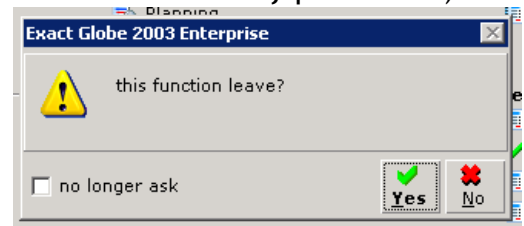
```
public class Dog {
    public static void bark() {
        System.out.println("wuff wuff wuff");
    }
}
```

```
...
Dog spot = new Dog();
spot.bark(); // Invoke bark on spot
wuff wuff wuff wuff
```

#49

Object Lingo

- “Apply a procedure” = “Invoke a method”
- We apply a procedure to parameters.
- We invoke a method on an object, and pass in parameters.
 - Inside a method you can also see the object itself (sometimes called the *self* parameter).



#50

Liberal Arts Trivia: Art History and American Literature

- Give the Renaissance master (or Ninja Turtle) associated with each work of art:



(b) Mona Lisa



(c) Pieta

(d) Transfiguration



#51

Liberal Arts Trivia: Polish History, Chemistry, and Physics

- This physicist and chemist of Polish upbringing and French citizenship was the first person honored with two Nobel prizes, the first woman to win a Nobel prize, and the first woman to serve as a professor at the University of Paris. The world's first studies into the treatment of cancers using radioactive isotopes were conducted under her direction.

#52

Liberal Arts Trivia: Cooking

- This Japanese delicacy is vinegared rice, usually topped with other ingredients, including fish. The dish as we know it today was invented as a fast food by Hanaya Yohei at the end of the Edo period (19th century) in Tokyo: it could be eaten on the road side or in a theatre using fingers or chopsticks. The basic idea can be traced back to 4th century BCE China as a preservative: the fermentation of the rice prevents the fish from spoiling.

#53

Dogs with Names

```
public class Dog {
    public String name; // Field (= State)
    public Dog(String n) { // Constructor
        name = n; // Initialize Field
    }
    public void bark() { // Method
        println(name + "says wuff!");
    } // Methods can see fields!
}

...
Dog myDog = new Dog("Spoticus"); // "new" calls Constructor, returns new object
myDog.bark(); // Invoke Method
Spoticus says wuff!
Dog yourDog = new Dog("Ginger"); // Not all objects have the same state!
yourDog.bark();
Ginger says wuff!
```



#54

Review: Making a Dog

```
public class Dog {
    Public void bark() {
        System.out.println("wuff wuff");
    }
}
...
Dog spot = new Dog(); // spot has type Dog
You must declare the type first!
```

#55

Java "Lists"

- Python has a built-in datatype for a list of fixed length. It is called an array [].


```
int [] myArray = {8,6,7};
println(myArray[0]);
1
println(myArray.length);
3
String [] myNames = {"Wes", "Weimer"};
println(myNames[1]);
Weimer
```

#56

Implementing square_each in Java

```
def square_each(lst):
    for i in range(len(lst)):
        lst[i] = lst[i] * lst[i] # imperative!
    • Let's do a literal translation of this into Java.
```

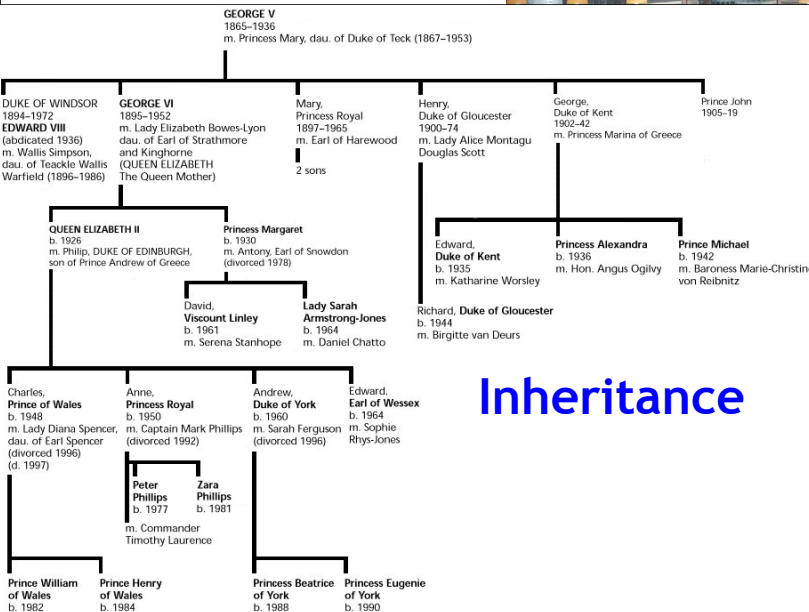


Java square_each

```
public static void square_each(int [] lst) {
    for (int i = 0; i < lst.length; i = i + 1) {
        lst[i] = lst[i] * lst[i] ; // still imperative!
    }
}
```

- Just like the previous one, this mutates lst.

#58



Hey, Scooby!

```
public class Dog {
    public Dog(n) { name = n; }
    public String name;
    public void bark() {
        println("wuff wuff");
    }
}
public class TalkingDog extends Dog {
    public void speak(String words) {
        println(name + " says " + words);
    }
} // inherits all Dog fields and methods
TalkingDog scooby = new TalkingDog("Scooby");
scooby.speak("solve the mystery!");
Scooby says solve the mystery!
```



#60

Subclasses

```
public class TalkingDog extends Dog {  
    public void speak(String words) {  
        println(name + " says " + words);  
    }  
} // inherits all Dog fields and methods
```

- TalkingDog is a **subclass** of Dog.
- Dog is the **superclass** of TalkingDog.
 - Every TalkingDog is also a Dog.
 - (But not vice-versa.)

#61

Every Dog Has Its Day

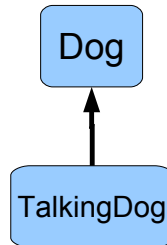
```
public class Dog {  
    public Dog(n) { name = n; }  
    public String name;  
    public void bark() {  
        println("wuff wuff");  
    }  
}  
public class TalkingDog extends Dog {  
    public void speak(String words) {  
        println(name + " says " + words);  
    }  
} // inherits all Dog fields and methods
```

```
Dog ginger = new Dog("Ginger");  
TalkingDog scooby = new Dog("Scooby");  
scooby.speak("snack!");  
Scooby says snack!  
ginger.speak("this won't work");  
Type Error  
scooby.bark();  
wuff wuff
```

#62

Speaking About Inheritance

- Inheritance is using the definition of one class to define another class.
- TalkingDog **inherits** from Dog.
- TalkingDog is a **subclass** of Dog.
- The **superclass** of TalkingDog is Dog.
- *These all mean the same thing!*



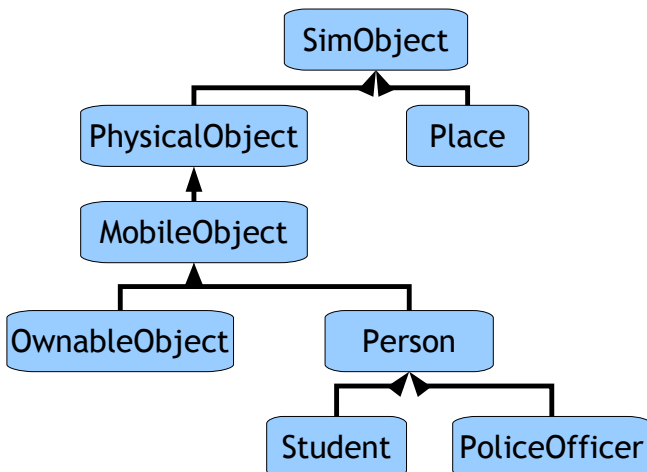
#63

Problem Set 6

- Make an adventure game by programming with objects.
- Many objects in our game have similar properties and behaviors, so we use inheritance.

#64

PS6 Classes



#65

Object-Oriented Terminology

- An **object** is an entity that packages state and procedures.
- The state variables that are part of an object are called **instance variables**.
- The procedures that are part of an object are called **methods**.
- We **invoke** (call) a method. The object itself and its fields are also visible in a method.
- **Inheritance** allows one class to refine and reuse the behavior of another.
- A **constructor** is a procedure that creates new objects (e.g., `public Dog() { ... }`).

#66

Charge

- Start PS6 early
 - PS6 is challenging
 - Opportunity for creativity
- Start thinking about PS9 Project ideas
 - If you want to do an “extra ambitious” project convince me your idea is worthy before Nov 10 (ps7 and 8) / Nov 17 (ps8)
 - Discuss ideas and look for partners **on the forum**

#67

Homework

- PS 5 due *soon*
- PS 6 due shortly thereafter

#68