Review of Java

… or maybe not

Have some *class*...

```
public _____ Foo _____ Bar _____ Comparable {
    /** name of this Foo */
    private String name;
    /** initialize a new Foo */
    public ____ Foo(String name) {
       ______________ = ________
    }
    public String toString() { return name; }

    // what other method is required?
}
```

Make the name *immutable*

```
public _____ Foo _____ Bar _____ Comparable {
    /** name of this Foo */
    private _________  String name;
    ...
```

immutable means you cannot change the value after it is set the first time.

Name these Primitives

32-bit whole numbers -9999 …. 0, 1, 0x10

64-bit whole numbers, written like 4L

true and false

'ก', 'ด', \u0420

2.98E+08

8-bit values 0, 1, …, 255 (not int values)

What is the result?

```
> char c = 'A';
> c  + 1
> (int) c
> (short) c
> (byte) c
> (char) 66
```

`(int) c` means "convert the value to an int".
This is called a *cast*.

You can use class name in a cast, too:

`(Character) c`

What is the result?

```
> c = 'ก' ;  (gau gai)

> (int) c

> ++c

> (char) c
```

Java uses Unicode for char and Strings,
but the output may not be readable if the output device doesn't use Unicode

Nerdy Math. What is the output?

```
> int n = 10;
> int x = n++;
> x    // what is the value?
> int y = ++n;
> y    // what is the value?
```

```
> n = 10;
> int y = n+++n+++n++;
> y
```

(a) 30   (b) 31   (c) 32   (d) 33   (e) Error

How are these different?

```
// 1 Billion + 2 Billion
> 1000000000 + 2000000000
> 1000000000L + 2000000000L
> 1E9 + 2E9

// in Java 7 you can write _ in numbers
> 1_000_000_000L + 2_000_000_000L
```

Bizarre Numbers

```
System.out.println( 12 );
```

```
System.out.println( 012 );
```

```
System.out.println( 0x12 );
```

```
System.out.println( 012 + 0x12 );
```

Which data type should you use for ...

```
// the day of the month
________  day = 13;  // 13 Jan 2015
// population of the world
________  worldPop = (7 billion) ;
// Bank account number
// example: 001230055555
________ accountNumber = . . .
```

How to Convert Primitive to Object?

A List (like ArrayList) can only contain *objects*.

How can we add *primitive values* (like int) to a List?

```
List mylist = new ArrayList( );
int n = 51651111;
mylist.add( n ); // Wait? How is this possible?
```

Try in BlueJ Codepad:

```
> List list = new ArrayList( );
> list.add( "hello" )
> list.add( 10 )    // what is being put in the List?
> list.get( 0 )
"apple"    (String)
> list.get( 1 )
<object reference>  (Integer)
```

Wrapper Classes

Primitive	**Wrapper**
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

```
double root = Math.sqrt( 2.0 );
Double d1 = new Double( root );
// same thing: automatic boxing
Double d2 = root;


// print as a string
out.println(  d2.toString( ) );
// static method to make a string
out.println( Integer.toString( 2 ) );
```

Methods to convert to/from String

```
int n = 29*31;
// convert n to a String
String product = Integer.toString(n);
// parse integer value of a String
String s = "123";
int m = Integer.___________ (s);
// parse double value of String s
double d = _________________________;
```

parseInt(string) versus valueOf(string)

What is the difference?

```
String s = "123";

Integer.parseInt( s )



Integer.valueOf( s )
```

Useful Constants in Numeric Wrapper classes

1. What is the largest "int" value?
2. What is the smallest "long" value?
3. What is the range (smallest, biggest) of double?

```
int maximum =

long minimum =

double minsize =

double maxsize =
```

What value is after the biggest value?

```
int n = Integer.MAX_VALUE;

n = n + 1;

System.out.println( n );

double d = Double.MAX_VALUE;

d = d + 1;

System.out.println( d );

d = d * 1.000001;

System.out.println( d );
```

Packages

- Java uses packages to organize classes.
- Packages reduce size of *name space* and avoid *name collisions* (like `Date` in `java.util` and `java.sql`).

Q: Which package contain these classes?

Java language core classes (Object, String, System, ...).
You never have to import this.

Classes for input and output, like InputStream, FileReader

Date classes and collections (List, ArrayList)

Utilities Scanner, Arrays,

Java Graphics frameworks (2 packages)

Packages

Where is …

String class java.lang - core classes of the java language

Scanner … java.util - utilities and Collections (ArrayList)

Date … java.util - date and time classes (expect java 8)

InputStream and FileReader … java.io

java.io – Input & Output classes

javax.swing - Swing graphics (also javax.swing.___)

java.awt - the AWT graphics framework

Identify each of these

Date

double

Double

System.out

System.out.println( )

System.nanoTime( )

Double.MAX_VALUE

java.lang.BigInteger

java.lang.*Comparable*

java.io

java.util.ArrayList

java.util.*List*

Is it a...

package

class

primitive type

attribute ("field") of object

static attribute of class

method (static or instance)

constant

interface

???

What is the output?

```
System.out.println( 3 + 4 );
```

```
System.out.println( "3" + 4 );
```

```
System.out.println( '3' + 4 );
```

```
System.out.println( 3 + "4" );
```

Bit Operations

```
> int a = 7;
> int b = 10;
> a & b
> a | b
> a ^ b
> a == b
> a = b
> a && b
```

Passing arguments to methods

```
public void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
// elsewhere in the code...
int n = 10;
int m = 20;
swap( m, n );
What is m?
```

Define a Person class

```
Person
------------------------------------------
- name: String

------------------------------------------
<<constructor>>
Person( name: String )
getName( ): String
setName( newname: String ): void
toString(): String
```

```java
Person p = new Person( "Pee" );
p.setName( "Nong" );
System.out.println( p.toString() );  // prints "Nong"
```

Passing arguments, again

```
public void swap(Person a, Person b) {
    Person temp = a;
    a = b;
    b = temp;
}
// elsewhere in the code...
Person m = new Person( "Meaw" );
Person n = new Person( "Nok" );
swap( m, n );
What is m.toString() ?
```

How about this?

```
public void swapName(Person a, Person b) {
    String temp = a.getName();
    a.setName( b.getName() );
    b.setName( temp );
}
// elsewhere in the code...
Person m = new Person( "Meaw" );
Person n = new Person( "Nok" );
swapName( m, n );
What is m.toString() ?
```

Difference between "==" and .equals?

```
> Double x = new Double(10);
> Double y = new Double(10);
> x == y
> String s = "yes";
> String t = "yes";
> s == t
> String u = new String("yes");
> s == u
> s.equals(u)
```

How to write equals( )

You should usually define **`equals( )`** like this:

```
public class Person {
     public boolean equals( Object other ) { … }
```

Not like this:

```
    public boolean equals( Person other ) { ... }
```

Javadoc

```
package ku.oop.contacts;
import java.util.List;
/**
 * A Person contains information about a
 * person including name and contact info.
 * @author Bill Gates
 * @since 2014.01.12
 */
public class Person {
   /** person's name, of course */
   private String name;
```

Write complete sentences, ending with period!

Method Javadoc

```
/**
 * Set the person's birthday.
 * @param birthday a date containing the
 *   person's birthday. Must not be null.
 */
public void setBirthday(Date birthday) {
   if (birthday == null)
     throw new IllegalArgumentException(
      "Read the javadoc, stupid!");
   .
   .
```

Method Javadoc with Return

```
/**
 * Withdraw money from the purse.
 * @param amount is amount to withdraw.
 * @return array of moneys withdrawn from
 *     purse, or null if can't perform the
 *     requested withdraw.
 */
public Money[] withdraw(double amount) {
   if (double <= 0.0) return null;
   .
   .
```

Bad Javadoc

```
/**
 * The Person class has name and birthday
 * @Bill Balmer
 * @Version 1.0
 */
package ku.oop.badcode;
public class Person {
    private String name;
    /**
     * get the firstname
     * @param k is the index of last char
     */
    public String getFirstname( ) {
        int k = name.indexOf(' ');
        return name.substring(0,k); // bug?
    }
```

Good Code has Documentation

- Use documentation to describe classes and methods.
- Describe what and why – not "how" which is obvious from the code.
- Describe rationale and logic which is not obvious from code.

```
// A useless comment
// sum elements in the array
int sum = 0;
for(int k=0; k<array.length; k++) {
    sum += array[k];
}
```

No Javadoc = No Credit

Generate Javadoc from your Code

3 ways:

- the `javadoc` command
- let Eclipse (or BlueJ or Netbeans or …) do it
- automatic build system, like Maven

JAR files

What is a JAR file?

Why use them?

How to create one?

WHERE ARE THE JDK CLASSES?

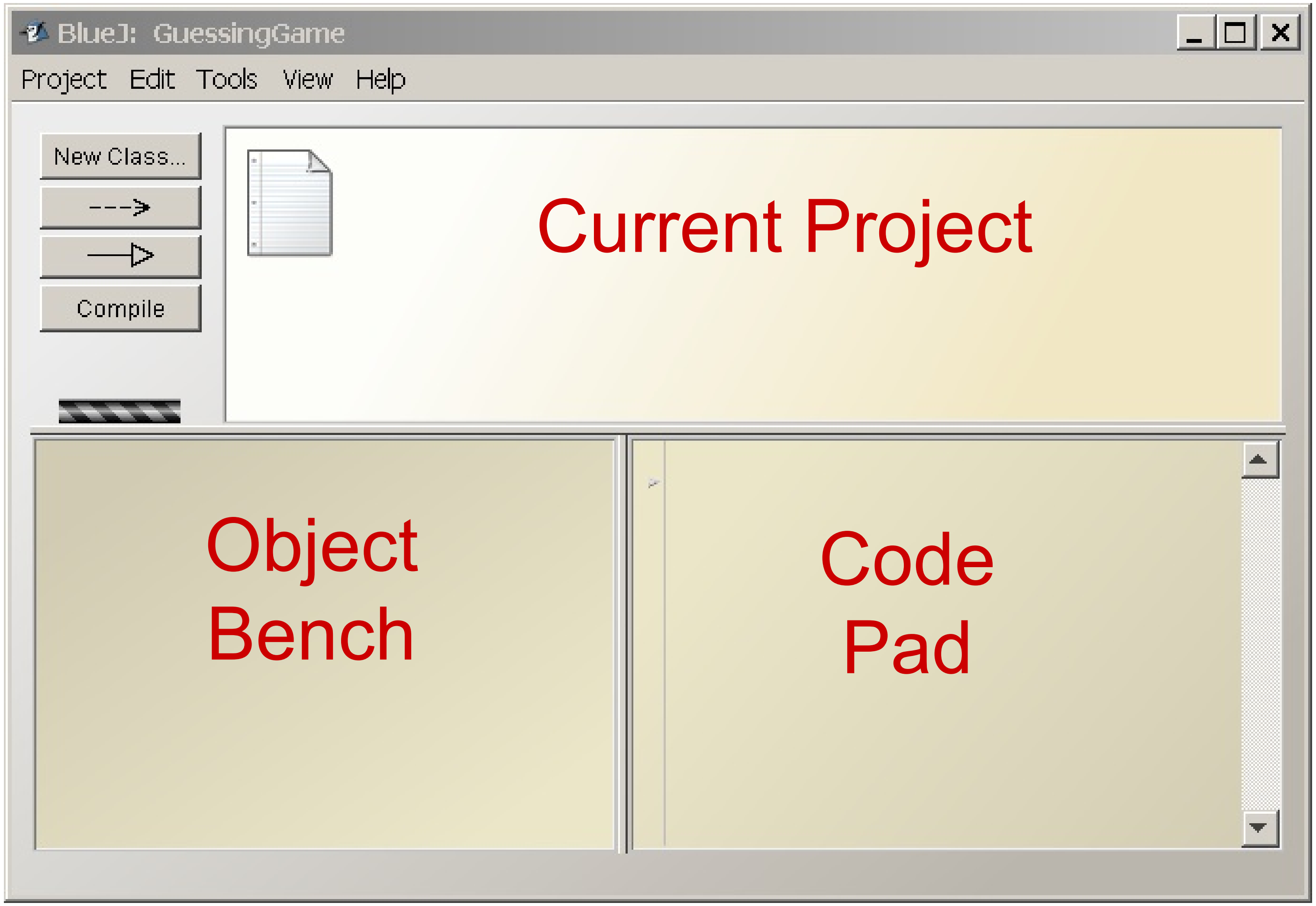
Classes in the Java SE API:

- 4,024 in java 7
- 3,793 in java 6
- 3,279 in java 5.0

Actually there are MORE classes that this – some classes are not documented in the API. And this number does not include *interfaces*.

These classes are on your computer (in the JDK). Where are they?

BlueJ IDE Layout


BlueJ: GuessingGame
Project Edit Tools View Help
New Class...
Compile
Current Project
Object Bench
Code Pad