abstract class —

- at least one abstract method
- can't be instantiated

abstract class Animal

(new Animal( )
won't work.)

abstract method -

- declared, but not "defined" (no code)

public void eat( );

identifies abstract method

interface - nothing but
- abstract methods
- public constants

describes related types of behavior

use of interfaces

```
public interface Eater{
    public void eat();
    public void gorge();
    public void nibble();
        .
        .
        .
}
```

I am permitted to have a class:

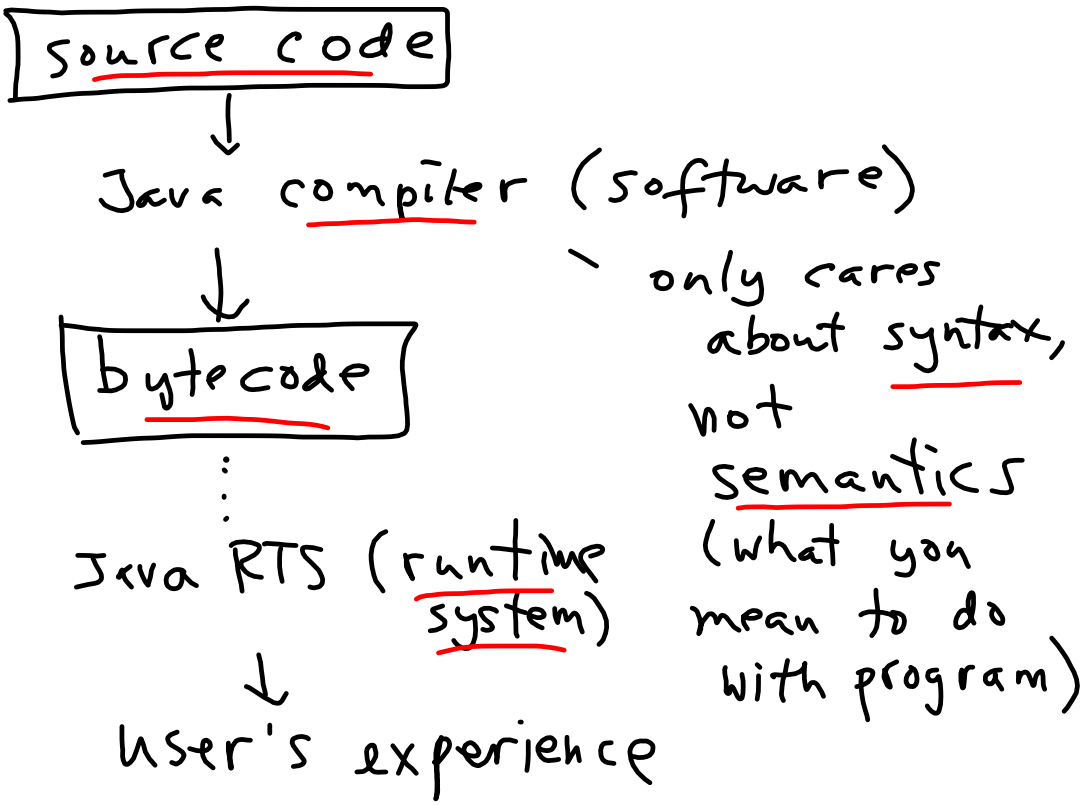
```
class GraySquirrel
        implements Eater{
        .
        .
        .
}
```

Now, I can do this:

Eater e = new GraySquirrel();

bit: a binary digit
either 0 or 1

byte: 8 bits
holds up to 2^8 values



constructor:

- special method that constructs an object
- has same name as its class
- returns reference to the constructed object
- invoked by new:

```
String s = new String("Hi!");
```

embedded – software put into everyday devices

extendible –

HAS-A vs IS-A relationships

HAS-A: object contains a reference to another object, as an instance variable

IS-A: class is a subclass of, or implements, a class or interface

IS-A:

class GraySquirrel extends Mammal

GraySquirrel IS-A Mammal

class BlueSquirrel implements Eater

BlueSquirrel IS-A Eater

HAS-A: class Target

```
{
    private Circle outer;
    private Target inner;
}
```

Target HAS-A Circle

Target HAS-A Target

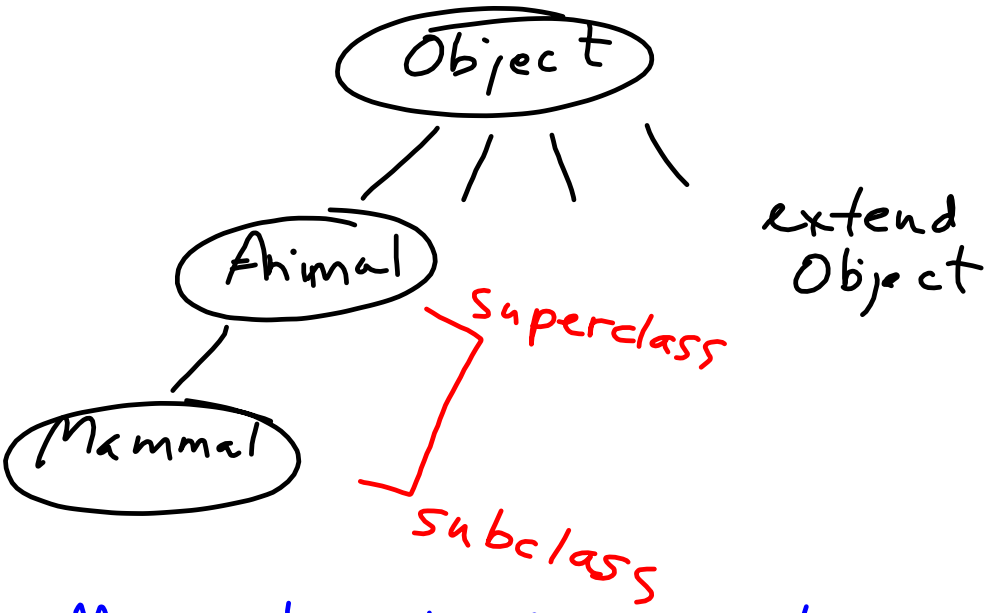
```
for (int i=0; i < arr.length; i++)
    for (int j=0; j < arr[0].length;
                        j++)
    {
        // code
    }
```

outer: i outer loop

inner: j inner loop

private: visibility only to class code, not client code

robust: capable of dealing with unexpected inputs or situations.



Mammal extends Animal

Legal Names (identifiers)

- can use letters, numbers, underscore
- can't start a name with a number
- constructor has same name as class

Conventions:

- methods, variables start with lowor-case, words inside name start with upper-case
- class name starts with upper-case

Strings

```
String s1 = "Hello, earth.",
       s2 = "Hello, World.",
       s3 = "Hello.";
```

```
s1.length()          13
s1.compareTo(s2)     positive int
s1.equals(s2)        false
s2.equals(s2)        true
s2 == s1             DO NOT USE!
s2.substring(3,7)    "lo, "
```

Math.random()

① expression returning a random int i such that 0 ≤ i ≤ 14 (# frog off, # frog on)

(int)(Math.random()*15.)

② return such that 5 ≤ i ≤ 21

5 + (int)(Math.random()*17.)

or (int)(5 + Math.random()*17.)

```
int i = -4, j = 6, k = 11;
double x = -2.7, y = 5.2;
```

j / k 0

k / j 1

k % j 5

(int)x -2

(int)(k / x) -4

11.0 / -2.7

-4.something

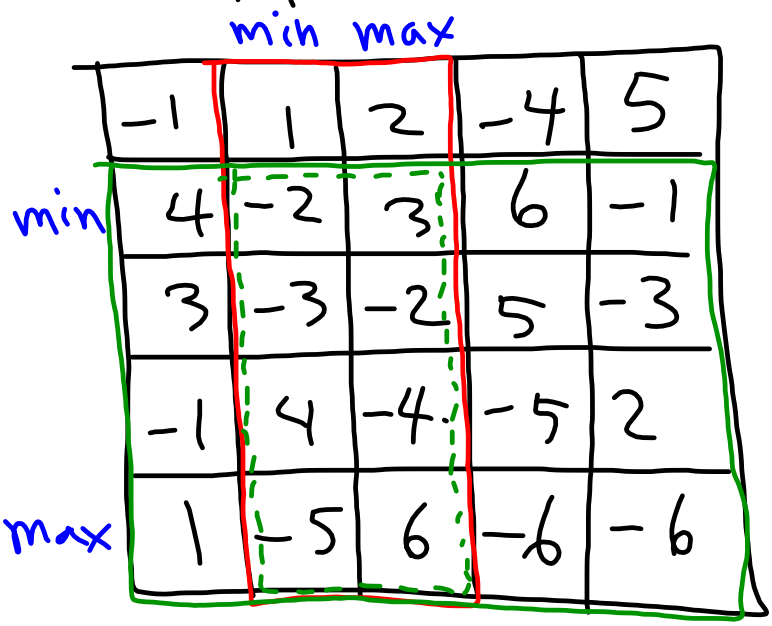
-4

passed to method:
- 2D array of int
- minimum column index
- maximum " "
- minimum row "
- maximum " "

returned from method:
- sum of absolute values of elements in these rows & columns only.

Hint: use Math.abs(i)
or ((conor): i>0?i:-i

illustration



return 29

```
public int getAbs(int minCol,
  int maxCol, int minRow, int maxRow
  int[][]arr)
{
  int absVal=0;
  for (int col=minCol;
    col <= maxCol; col++)
  for(int row=minRow;
      row<=maxRow;
      row++)
  absVal = absVal +
  Math.abs(
  arr[row][col]);
  return absVal;
}
```

```
absVal += Math.abs(arr[row][col]);
```

Final: categories

1. Vocab 19 questions
2. declarations & initializations 5 questions.
3. fill-in-blank method headers
4. names - valid or not? 1 multi-part question
5. Math.random() to return random int 2 questions
6. String exercises 1 multi-part
7. Expressions - type & value - 1 multi-part
8. 2D array method