A programmer heads out to the store. Their spouse says, "while you're out get some milk."

The programmer never came home.

Announcements

- Jotto – Due September 24
- APT Set 3 – Due September 26
  - It's posted
- Midterm 1 – October 2

Last time

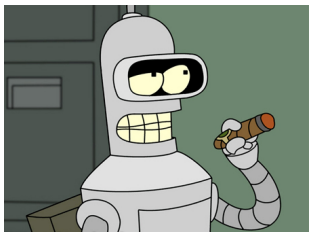
- HashTable
  - array of fixed size

  - with a key to each location
  - each key is mapped to an index in the table

0	
1	joe 31
2	
3	mary 43
4	sam 14
5	
6	
7	
8	
9	

Today

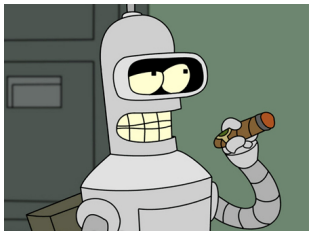
- extends
- Object
- abstract
- interface



with robots

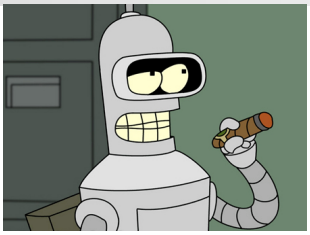
Today

- extends
- Object
- abstract
- interface



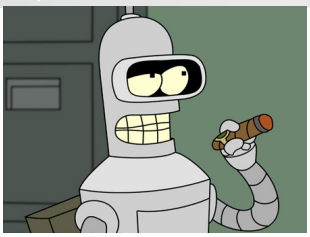
with robots

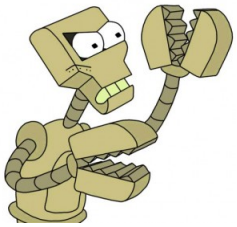
extends



```
class BendingRobot

doBending()
useElectricity()
```

extends

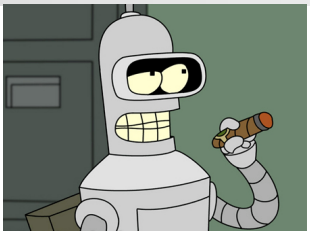


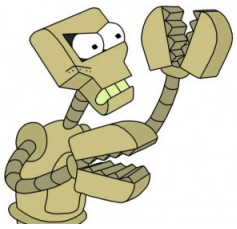


```
class BendingRobot

doBending()
useElectricity()
```

```
class ClampingRobot

doClamping()
useElectricity()
```


extends



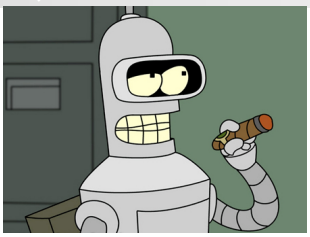


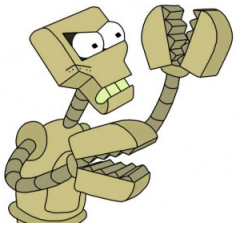
```
class BendingRobot

doBending()
useElectricity()
```

```
class ClampingRobot

doClamping()
useElectricity()
```

```
class EvilSantaRobot

doPunishNaughtyChildren()
useElectricity()
```

extends





```
class BendingRobot
doBending()
useElectricity()
```

```
class ClampingRobot
doClamping()
useElectricity()
```

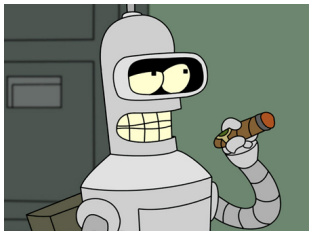
```
class EvilSantaRobot
doPunishNaughtyChildren()
useElectricity()
```


extends

```
class Robot{
  public void useElectricity{

  }

  //other common robot functions

}
```

extends

```
class BendingRobot extends Robot{

   doBending(){}
   useElectricity(){}

}
```



- BendingRobot inherits useElectricity from Robot.

extends

```
class GoldBendingRobot extends
BendingRobot{

   doSuperBending(){}
   doBending(){}
   useElectricity(){}

}
```

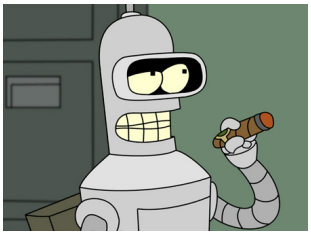
extends

- BendingRobot – Superclass
  - GoldBendingRobot – Subclass
  - Subclass inherits from superclass

```
GoldBendingRobot goldBot = new GoldBendingRobot();
goldBot.doBending();
goldBot.doSuperBending();
```


extends

- BendingRobot – Superclass
  - GoldBendingRobot – Subclass
  - Superclass does not inherit from subclass



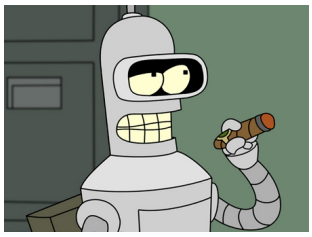
```
BendingRobot someBender = new GoldBendingRobot();
someBender.doBending();
someBender.doSuperBending(); //NOT ALLOWED
```

extends

- A extends B – A inherits all functions and variables from B
- Subclass A can be used any place superclass B can.
- Subclass A can "override" functions in superclass B
- Always use the most general type possible

Today

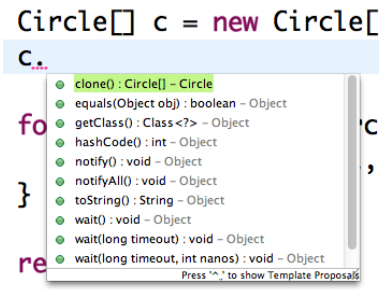
- extends
- Object
- abstract
- interface



with robots

Object

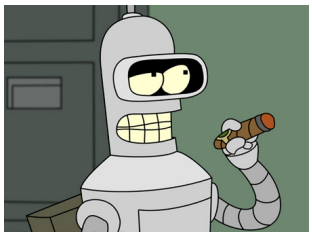
- Object (Java class) – superclass of your class
- All objects inherit
  - .equals()
  - .toString()
  - .hashCode()



- You can Override superclass with your own code!

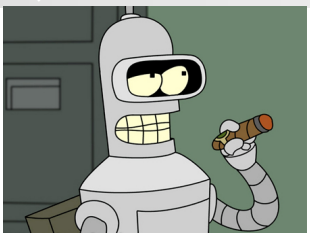
Today

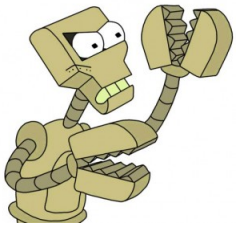
- extends
- Object
- abstract
- interface



with robots

abstract





```
class BendingRobot
doBending()
useElectricity()
```

```
class ClampingRobot
doClamping()
useElectricity()
```

```
class EvilSantaRobot
doPunishNaughtyChildren()
useElectricity()
```


abstract

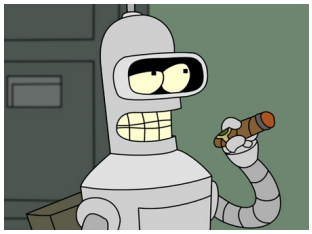
```
abstract class GenericRobot{
    abstract public void useElectricity();
        //all robots use electricity
        //but it may be different!!!!

    public void beep(){
        System.out.println("Beep!");
        //this is the same for all robots!!!!
    }
}
```

abstract

```
1  class BendingRobot extends GenericRobot
   {
2      //must implement abstract methods
3      public void useElectricity() {
4           //your code goes here
5      }
6  }
```


abstract

```
1  //doesn't matter what kind of Robot
2  GenericRobot myRobot = new BendingRobot();
3  BendingRobot bendRobot = new BendingRobot();
4  myRobot.beep();
5  myRobot.useElectricity();
6  GenericRobot otherBot = new GenericRobot(); //
   NOT ALLOWED
```

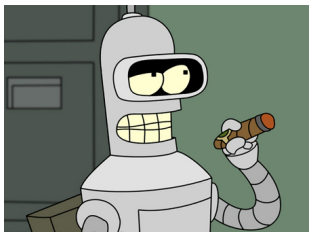


abstract

- *abstract* superclass contains *abstract* functions
- Subclass A of abstract superclass B must implement the abstract functions
- Can have variables of abstract superclass type, but cannot create objects of abstract superclass (can never use new)

Today

- extends
- Object
- abstract
- interface



with robots

interface

```
abstract class Robot{
    abstract public void useElectricity();
        //all robots use electricity
        //but it may be different!!!!

    public void beep(){
        System.out.println("Beep!");
        //this is the same for all robots!!!!
    }
}
```

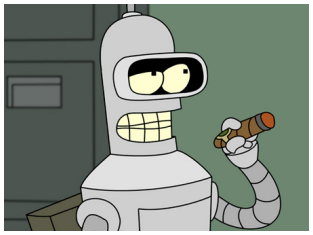

interface

```
interface MoveRobot{
        void moveForward();
}
```

*all of our methods are abstract

interface

```
class BendingRobot implements
MoveRobot{

   doBending(){}
   moveForward(){}
}
```

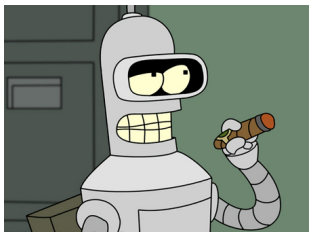


- BendingRobot must implement methods from MoveRobot.

9/17/13 27

interface

```
class BendingRobot extends Robot
implements MoveRobot{

   doBending(){}
   moveForward(){}
   useElectricity(){}
}
```



- BendingRobot must implement methods from MoveRobot, but inherits methods from Robot.

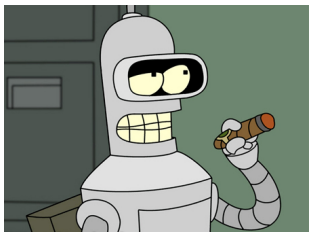
9/17/13 28

interface

- Similar to a superclass, but has no implemented methods
- You implement an interface in the same way you extend a superclass
- You can implement many interfaces, but only extend one superclass

Today

- extends
- Object
- abstract
- interface



with robots