CS61BL - Inheritance

CS61BL Summer 2009
6/30/2009

Important Dates

- Project 1 released
  - Tuesday 6/30/2009 – 10pm
- Project 1 - Check-off
  - Thursday 7/02/2009 ready BEFORE lab
- Review Session
  - Sunday 7/05/2009 – 306 Soda 1-4pm
- Midterm 1
  - Tuesday 7/07/2009 – 10 Evans 5-6pm
- Project 2 released
  - Thursday 7/09/2009
- Project 1 due
  - Monday 7/13/2009 – 10pm

Estimating Time

- Most people grossly underestimate how long it takes to program (not thinking about time to debug and test)
- You appear incompetent at programming (not true) if you can’t estimate how long something will take.
- This is one of the hardest things to learn!!! Start to practice now! For each task estimate how long it will take and keep track of how wrong you were

Test Driven Development

```java
static void beAGoodProgrammer() {
    while (true) // for each method
    {
        writeTest();
        writeCode();
        writeAFewMoreTests();
        writeCode();
    }
}
```
Why do Test Driven Development?

- It is a way to brainstorm what your method is going to do.
- It is hard to write test cases
- Write your test cases before you’re biased by how your code is written

Inheritance

- A Dog is an Animal
- Dog extends Animal
- A Cat is an Animal
- Cat extends Animal
- A Poodle is an Animal
- Poodle extends Dog
- A Dalmatian is an Animal
- Dalmatian extends Dog

```java
public class Animal {
    protected int age = 0;
    public void becomeOlder() {
        this.age ++;
    }
    public void eat() {
        System.out.println("Animal eat");
    }
}

Animal myAnimal = new Animal();
myAnimal.becomeOlder();
myAnimal.eat();
```

```java
public class Dog extends Animal {
    protected boolean bites = true;
    public void eat() {
        System.out.println("Dog eat");
    }
    public void bark() {
        System.out.println("woof");
    }
}

Dog myDog = new Dog();
myDog.eat();
myDog.bark(); // ↓ does this work? ___
myDog.becomeOlder();
```
```java
public class Animal {
    protected int age = 0;
    public void eat() {
        System.out.println("Animal eat");
    }
}

Animal myAnimal = new Animal();
myAnimal.bark();

↑Does this work????
```

---

**Subclasses (Children) Can**

- Declare new instance/class variables
- Declare new methods
- Override old implementations with new implementations
- Access instance/class variables of the superclass (parent) *
- Access methods of the superclass*

* only public, protected or default ones

---

**Inheritance Puzzles**

Dog d = new Animal();

Does this work? ______

No it doesn't compile, if I want a Dog, an Animal will be NOT be acceptable because I might want to call a method that only a Dog can do!

---

**Inheritance Puzzles**

Animal a = new Dog();

Does this work? ______

YES it compiles and runs - If I want an Animal, a Dog will be acceptable because everything an Animal can do a Dog can do too!

Animal a = (Animal) new Dog();

---

**Inheritance Puzzles**

Animal a = new Dog();
Dog d = a;

Does this work? ______

No it doesn't compile. The compiler doesn't know it is really a Dog!
but we can PROMISE the compiler that it is with a cast

---

**Inheritance Puzzles**

Animal a = new Animal();
Dog d = (Dog) a;

Does this work? ______

YES and NO - The compiler trusts your PROMISE (cast) but then if you run your program it has a RUN TIME ERROR because a is not a Dog
Inheritance Puzzles

Animal a = new Cat();
Dog d = (Dog) a;

Does this work? _______

YES and NO. It compiles, but then when we try to do the cast we get a RUN TIME ERROR, before we even try to call any dog methods on a.

Inheritance Puzzles

Animal a = new Dog();
a.bark();

Does this work? _______

NO it does not compile - The compiler thinks it is an Animal, and an Animal does not have a method bark.
Dog d = (Dog) a; d.bark();
((Dog) a).bark();

Inheritance Puzzles

Animal a = new Dog();
a.eat();

Does this work? _______

YES - The compiler is happy - an Animal has an eat() method.
Which method is called the Animal eat() or the Dog eat()?

Inheritance Puzzles

Animal a = new Dog();
a.eat();

Which method is called the Animal eat() or the Dog eat()?