Announcements

- A3 will be available on Piazza tomorrow. Refer often to the Piazza FAQ Note for A3.
- Please read the assignment A1 FAQ Notes on the Piazza before asking a question. It might already be answered.

Assignment A3: Doubly linked Lists

Idea: maintain a list (2, 5, 7) like this:

This is a singly linked list

To save space we write names like a6 instead of N@35abcd00

How to insert a node at the beginning (2, 5, 7)

How to remove successor of a node in the middle (2, 5, 8, 7)

Assignment A3: Use an inner class

Inside-out rule: Objects of Cin can reference components of the object of C in which they live.
Assignment A3: Generics

```java
public class LinkedList {
    void add(Object elem) {...}
    Object get(int index) {...}
}
```

Values of linked list are probably of class Object

```java
public class LinkedList<E> {
    void add(E elem) {...}
    E get(int index) {...}
}
```

You can specify what type of values

```java
ns = new LinkedList<Integer>();
ns.add("Hello"); // error
ns.add(5);
String s = ns.get(0); // error
int n = ns.get(0);
```

```java
ss = new LinkedList<String>();
ss.add("Hello");
ss.add(5); // error
String s = ss.get(0);
int n = ss.get(0); // error
```

Overview ref in text and JavaSummary.pptx

- Quick look at arrays slide 50-55
- Casting among classes C.33-C.36 (not good) slide 34-41
- Consequences of the class type slide 34-41
- Operator instanceof slide 40
- Function equals slide 37-41

Homework. Learn about while/ for loops in Java. Look in text.

while (<bool expr>) { ... } // syntax

for (int k= 0; k < 200; k= k+1) { ... } // example

Classes we work with today

- Work with a class Animal and subclasses like Cat and Dog
- Put components common to animals in Animal
- Object partition is there but not shown

Add function toString() to String class

```java
class Animal {
    @Override
    public String toString() { return "Animal object"; }
}
```

```java
class Cat {
    @Override
    public String toString() { return "Cat object"; }
}
```

```java
class Dog {
    @Override
    public String toString() { return "Dog object"; }
}
```


From an Animal variable, can use only methods available in class Animal

- `a.getWeight()` is obviously illegal. The class won’t compile.
  - When checking legality of a call like `a.getWeight(...)`, since the type of `a` is Animal, function `getWeight` must be declared in Animal or one of its superclasses.

Another example

- Type of `v[0]`: Animal
  - Should this call be allowed? Should program compile?
    - `v[0].getWeight()` is not in class Animal or Object. Calls are illegal, program does not compile:
      - `v[0].getWeight()` v[k].getWeight()

View of object based on the type

- Each element `v[k]` is of type Animal. From `v[k]`, see only what is in partition Animal and partitions above it.
  - Components are in lower partitions, but can’t see them
    - `v[0].getWeight()` v[k].getWeight()

Rule for determining legality of method call

- Rule: `c.m(...)` is legal and the program will compile ONLY if method `m` is declared in `C` or one of its superclasses
  - `c.m()` must be declared in one of these classes

Suppose `a0` contains an object of a subclass Cat of Animal. By the rule below, `a0.getWeight(...)` is still illegal. Remember, the test for legality is done at compile time, not while the program is running. …

The same object `a0`, from the viewpoint of a Cat variable and an Animal variable

- `c.getWeight()` is legal
- `a0.getWeight()` is illegal because `getWeight` is not available in class Animal

From an Animal variable, can use only methods available in class Animal

- `a.getWeight()` is obviously illegal. The class won’t compile.
  - When checking legality of a call like `a.getWeight(...)`, since the type of `a` is Animal, function `getWeight` must be declared in Animal or one of its superclasses.

Suppose `a0` contains an object of a subclass Cat of Animal. By the rule below, `a0.getWeight(...)` is still illegal. Remember, the test for legality is done at compile time, not while the program is running. …
Casting objects

You know about casts like:

(int) (5.0 / 7.5)
(doubler) 6

double d = 5; // automatic cast

Discuss casts up/down class hierarchy.

Animal h = new Cat("N", 5);
Cat c = (Cat) h;

A class cast doesn’t change the object. It just changes the perspective –how it is viewed!

Explicit casts: unary prefix operators

Rule: an object can be cast to the name of any partition that occurs within it — and to nothing else.

a0 can be cast to Object, Animal, Cat.

Attempt to cast it to anything else causes an exception

These casts don’t take any time. The object does not change. It’s a change of perception.

Implicit upward cast

Variable h is created. a1 is cast up to class Animal and stored in h.

Upward casts done automatically when needed

Explicit downward cast

Type of h is Animal. Syntactic property. Determines at compile-time what components can be used; those available in Animal

Component used from h

Components used from h

By overriding rule, calls toString() in Dog partition

(Dog) ob leads to runtime error. Don’t try to cast an object to something that it is not!
public class Cat extends Animal {
    private float weight;
    /** return true iff ob is a Cat and its
    * fields have same values as this */
    public boolean equals(Object ob) {
        if (! (ob instanceof Cat)) return false;
        // { h is a Cat }
        if (! super.equals(ob)) return false;
        Cat c = (Cat) ob; // downward cast
        return weight == c.getWeight();
    }
}

Opinions about casting

- Use of instanceof and downcasts can indicate bad design
  - **DONT:**
    - if (x instanceof C1) do thing with (C1) x
    - else if (x instanceof C2) do thing with (C2) x
    - else if (x instanceof C3) do thing with (C3) x
  - **DO:**
    - x.do() (do thing overridden in C1, C2, C3)

- But how do I implement equals()? That requires casting