

1

CS/ENGRD 2110
FALL 2015

Lecture 6: Consequence of type, casting; function equals
<http://courses.cs.cornell.edu/cs2110>

Announcements

- A3 now available on CMS and Piazza. Refer often to the Piazza FAQ Note for A3
- Please read the assignment 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

Easy to insert a node at the beginning!

(2, 5, 7)

(8, 2, 5, 7)

Easy to remove a node if you have its predecessor!

(2, 5, 8, 7)

(2, 5, 7)

Assignment A3: Use an inner class

```
public class LinkedList {
    private int x;
    public void m(int y) { ... }
    private class CI {
    }
}
```

Inside-out rule: Objects of CI can reference components of the object of C in which they live.

In addition: methods of C can reference private components of CI

Assignment A3: Generics

```

public class LinkedList {
}

public class LinkedList<E> {
}

new LinkedList<Integer>(...)
new LinkedList<String>(...)
new LinkedList<JFrame>(...)
    
```

Values of linked list are probably of class Object

You can specify what type of values

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
    
```

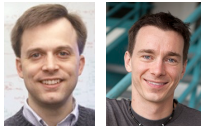
Big Picture: Type Systems

Object types in Java

- Arrays
- Subtypes
- Method resolution
- Casts
- Binary methods

Cornell Research

- Polyglot Compiler
- Object initialization
- Information-flow
- Pattern matching
- Decidability



Andrew Myers Ross Tate

Classes we work with today

class hierarchy:

```

graph TD
    Object --> Animal
    Animal --> Dog
    Animal --> Cat
    
```

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

a0	age 5	Animal	Animal(String, int)	isOlder(Animal)
a1	age 6	Animal	Animal(String, int)	isOlder(Animal)
		Cat(String, int)	Cat	getNoise() toString() getWeight()
		Dog(String, int)	Dog	getNoise() toString()

Animal[] v= new Animal[3];

declaration of array v

Create array of 3 elements

Assign value of new-exp to v

Assign and refer to elements as usual:

```

v[0]= new Animal(...);
...
a= v[0].getAge();
    
```

Sometimes use horizontal picture of an array:

v	0	1	2
	null	null	null

Which function is called?

Which function is called by v[0].toString() ?

Remember, partition Object contains toString()

Bottom-up or overriding rule says function toString in Cat partition

a0	age 5	Animal	Animal(String, int)	isOlder(Animal)
a1	age 6	Animal	Animal(String, int)	isOlder(Animal)
		Cat(String, int)	Cat	toString() toNoise() getWeight()
		Dog(String, int)	Dog	toString() toNoise()

Consequences of a class type

13

Animal[] v; declaration of v. Also means that each variable v[k] is of type Animal

The type of v is Animal[]
The type of each v[k] is Animal
The type is part of the syntax/grammar of the language. Known at compile time.

0	1	2
a0	null	a1

Animal objects

As we see on next slide, the type of a class variable like v[k] determines what methods can be called

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

14

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.

a0	Animal
age 5	Animal(String, int)
	isOlder(Animal)

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

15

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

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.

a0	Animal
age 5	Animal(String, int)
	isOlder(Animal)
Cat(String, int)	Cat
getNoise() toString()	getWeight()

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

16

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

c.getWeight() is legal a.getWeight() is illegal

because getWeight is not available in class Animal

c	a0	Animal
	age 5	Animal(String, int)
		isOlder(Animal)
	Cat(String, int)	Cat
	getNoise() toString()	getWeight()

Rule for determining legality of method call

17

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

m(...) must be declared in one of these classes

a0	Object
	...
	...
	C

Another example

18

Type of v[0]: Animal

Should this call be allowed? Should program compile?

v[0].getWeight()

Should this call be allowed? Should program compile?

v[0].getWeight()

v	a0	Animal
	age 5	Animal(String, int)
		isOlder(Animal)
	Cat(String, int)	Cat
	getNoise() toString()	getWeight()

a1	Animal
age 6	Animal(String, int)
	isOlder(Animal)
Dog(String, int)	Dog
getNoise() toString()	getWeight()

View of object based on the type

19

Each element `v[k]` is of type `Animal`. From `v[k]`, see only what is in partition `Animal` and partitions above it.

`getWeight()` not in class `Animal` or `Object`. Calls are illegal, program does not compile:
`v[0].getWeight() v[k].getWeight()`

Components are in lower partitions, but can't see them

v		
0	1	2
a0	null	a1

Animal

a0

age 5	Animal
Animal(String, int)	
isOlder(Animal)	
Cat(String, int)	Cat
getNoise() toString() getWeight()	

a1

age 6	Animal
Animal(String, int)	
isOlder(Animal)	
Dog(String, int)	Dog
getNoise() toString()	

Casting up class hierarchy

20

You know about casts like

```
(int) (5.0 / 7.5)
(double) 6
double d = 5; // automatic cast
```

```

graph TD
    Object --> Animal
    Animal --> Dog
    Animal --> Cat
    
```

a0

age 5	Animal
Animal(String, int)	
isOlder(Animal)	
Cat(String, int)	Cat
getNoise() toString() getWeight()	

a1

age 6	Animal
Animal(String, int)	
isOlder(Animal)	
Dog(String, int)	Dog
getNoise() toString()	

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

21

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

`a0` maybe cast to `Object`, `Animal`, `Cat`.
 An attempt to cast it to anything else causes an exception

```
(Cat) c
(Object) c
(Animal) (Animal) (Cat) (Object) c
```

a0

equals() ...		Object
age 5	Animal	
Animal(String, int)		
isOlder(Animal)		
Cat(String, int)	Cat	
getNoise() toString() getWeight()		

c a0

Cat

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

Implicit upward cast

22

```
public class Animal {
    /** = "this Animal is older than h" */
    public boolean isOlder(Animal h) {
        return age > h.age;
    }
}
```

Call `c.isOlder(d)`

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

Upward casts done automatically when needed

h a1

Animal

c a0

Cat

d a1

Dog

a0

age 5	Animal
Animal(String, int)	
isOlder(Animal)	
Cat(String, int)	Cat
getNoise() toString() getWeight()	

a1

age 6	Animal
Animal(String, int)	
isOlder(Animal)	
Dog(String, int)	Dog
getNoise() toString()	

Example

23

```
public class Animal {
    /** = "this is older than h" */
    public boolean isOlder(Animal h) {
        return age > h.age;
    }
}
```

h a1

Animal

a1

age 6	Animal
Animal(String, int)	
isOlder(Animal)	
Dog(String, int)	Dog
getNoise() toString()	

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

If a method call is legal, the overriding rule determines which method is called

Components used from h

24

```
public class Animal {
    /** = "this is older than h" */
    public boolean isOlder(Animal h) {
        return age > h.age;
    }
}
```

h a1

Animal

a1

age 6	Animal
Animal(String, int)	
isOlder(Animal)	
Dog(String, int)	Dog
getNoise() toString()	

`h.toString()` OK —it's in class `Object` partition

`h.isOlder(...)` OK —it's in `Animal` partition

`h.getWeight()` ILLEGAL —not in `Animal` partition or `Object` partition

By overriding rule, calls `toString()` in `Dog` partition

Explicit downward cast

25

```

public class Animal {
    // If Animal is a Cat, return its weight;
    // otherwise, return 0.
    public int checkWeight(Animal h) {
        if ( !
            )
            return 0;
        // { h is a Cat }
        Cat c= (Cat) h ; // downward cast
        return c.getWeight();
    }

```

a0
age 5 Animal
Animal(String, int)
isOlder(Animal)
Cat(String, int) Cat
getNoise() toString()
getWeight()

h a0
Animal

(Dog) h leads to runtime error.
Don't try to cast an object to something that it is not!

Operator instanceof, explicit downward cast

26

```

public class Animal {
    // If Animal is a cat, return its weight;
    // otherwise, return 0.
    public int checkWeight(Animal h) {
        if ( ! (h instanceof Cat) )
            return 0;
        // { h is a Cat }
        Cat c= (Cat) h ; // downward cast
        return c.getWeight();
    }

```

a0
age 5 Animal
Animal(String, int)
isOlder(Animal)
Cat(String, int) Cat
getNoise() toString()
getWeight()

h a0
Animal

<object> instanceof <class>
true iff object is an instance of the class —if object has a partition for class