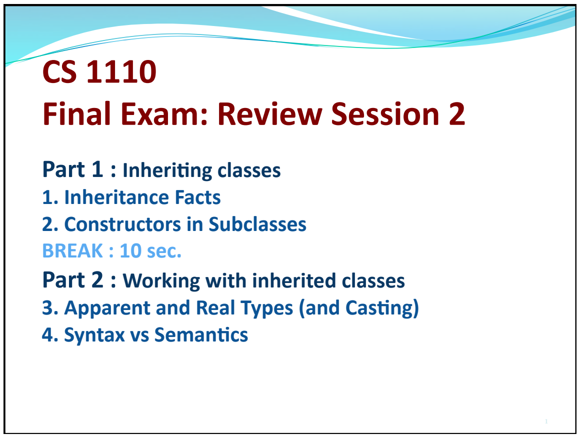


CS 1110
Final Exam: Review Session 2

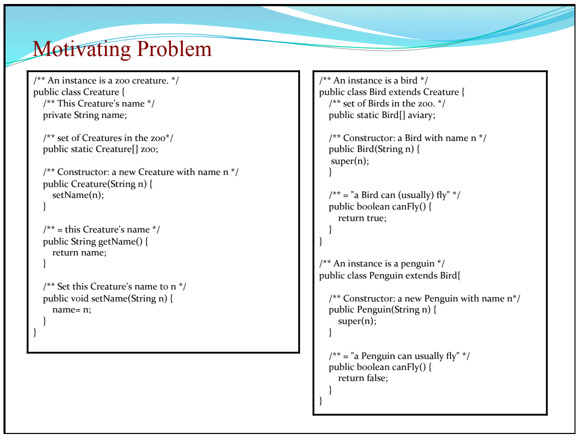
Part 1 : Inheriting classes

1. Inheritance Facts
2. Constructors in Subclasses

BREAK : 10 sec.

Part 2 : Working with inherited classes

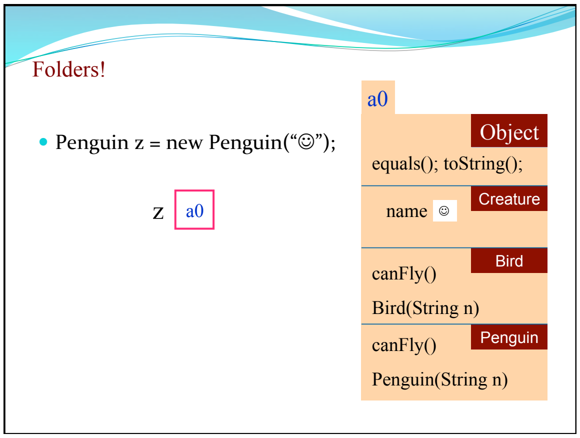
3. Apparent and Real Types (and Casting)
4. Syntax vs Semantics



Motivating Problem

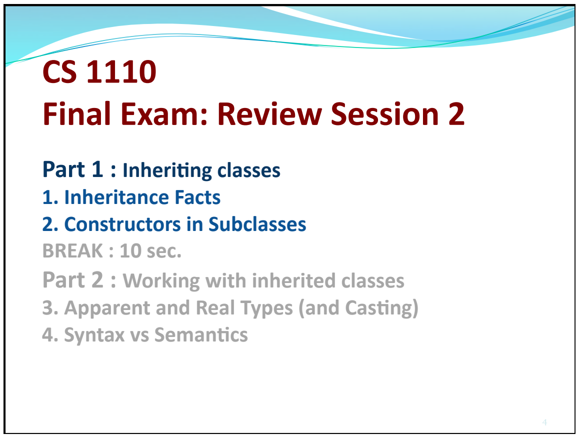
```
/** An instance is a zoo creature. */
public class Creature {
   /** This Creature's name */
   private String name;

   /** set of Creatures in the zoo*/
   public static Creature[] zoo;

   /** Constructor: a new Creature with name n */
   public Creature(String n) {
      setName(n);
   }

   /** = this Creature's name */
   public String getName() {
      return name;
   }

   /** Set this Creature's name to n */
   public void setName(String n) {
      name= n;
   }
}
```

```
/** An instance is a bird */
public class Bird extends Creature {
   /** set of Birds in the zoo. */
   public static Bird[] aviary;

   /** Constructor: a Bird with name n */
   public Bird(String n) {
    super(n);
   }

   /** = "a Bird can (usually) fly" */
   public boolean canFly() {
      return true;
   }
}

/** An instance is a penguin */
public class Penguin extends Bird{

   /** Constructor: a new Penguin with name n*/
   public Penguin(String n) {
      super(n);
   }

   /** = "a Penguin can usually fly" */
   public boolean canFly() {
      return false;
   }
}
```



Folders!

- Penguin z = new Penguin("☺");



CS 1110
Final Exam: Review Session 2

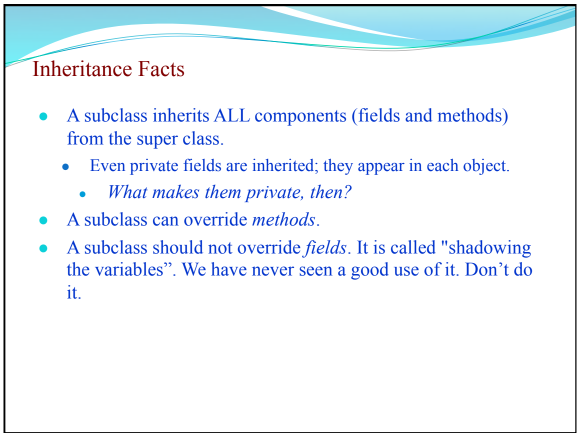
Part 1 : Inheriting classes

1. Inheritance Facts
2. Constructors in Subclasses

BREAK : 10 sec.

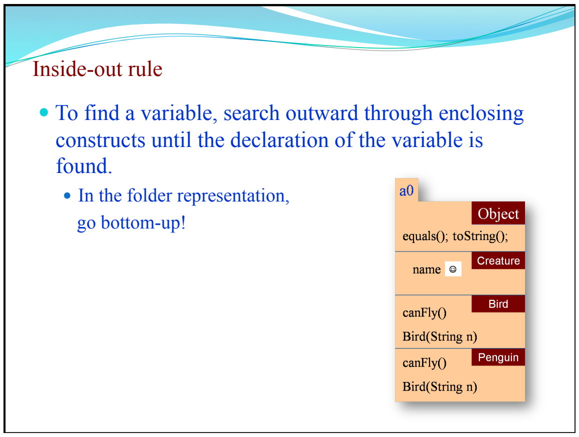
Part 2 : Working with inherited classes

3. Apparent and Real Types (and Casting)
4. Syntax vs Semantics



Inheritance Facts

- A subclass inherits ALL components (fields and methods) from the super class.
  - Even private fields are inherited; they appear in each object.
    - *What makes them private, then?*
- A subclass can override *methods*.
- A subclass should not override *fields*. It is called "shadowing the variables". We have never seen a good use of it. Don't do it.



Inside-out rule

- To find a variable, search outward through enclosing constructs until the declaration of the variable is found.
  - In the folder representation, go bottom-up!

Constructors in subclasses

- The following can only appear as the first statement in a constructor:
  - this(...);   // call another constructor in this class
  - super(...); // call a constructor in the super class
- If there is no explicit constructor call in a constructor, Java inserts super();.
  - *What if the super class does not have a default constructor?*

Note: If there is no explicit constructor declared in a class, Java inserts a default constructor.

CS 1110
Final Exam: Review Session 2

Part 1 : Inheriting classes
1. Inheritance Facts
2. Constructors in Subclasses

BREAK : 10 sec.

Part 2 : Working with inherited classes
3. Apparent and Real Types (and Casting)
4. Syntax vs Semantics

CS 1110
Final Exam: Review Session 2

Part 1 : Inheriting classes
1. Inheritance Facts
2. Constructors in Subclasses
BREAK : 10 sec.

Part 2 : Working with inherited classes
3. Apparent and Real Types (and Casting)
4. Syntax vs Semantics

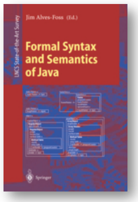
Apparent & Real Types (and Casting)

- Real type of a variable
  - The type of the object it's referring to.
- Apparent type of a variable:
  - The type of the variable…-_-
  - Used to tell if a reference is legal (if not, program won't compile.).
    - v.field / v.method(...) is legal only if field / method() is defined in or inherited by the apparent class of v.
- Casting: changing of the apparent type
  - *What are the restrictions?*

Syntax VS Semantics

- The validity of a statement should be checked by:
  - SYNTAX (grammar; rules for legal programs)
  - SEMANTICS (meaning; how legal programs are executed).
- Ex) Assess the following statements:
  - a == b
  - a.equals(b)

What makes one correct, but not the other?


Jim Alves-Foss (Ed.)
Formal Syntax and Semantics of Java
LNCS State-of-the-Art Survey
Springer

Running Example!

- ***Dr Java ☺***