CS 3710 Vocational Languages

Nov 23, 2012

Traits vs. Abstract Classes

• Similarities:
  – Both are abstract, which means that they can not be instantiated (new).
  – Both can define abstract and concrete instance variables.
  – Both can define abstract and concrete methods.

• Differences:
  – A class can only extend one other class, which can be either abstract or concrete: single inheritance.
  – A class can extend multiple traits

Extend Traits vs. Extend Classes

• Reuse: Extend a class and extend a trait make the methods and instance variables of the class and trait available for the sub-class.
• Extend a Class: all methods and instance variables of the extended class are inherited by the class
• Extend a Trait: all methods and instance variables of the extended Trait are “mixed in” with the class, i.e. the compiler adds the mixed-in methods/instance variables to the class.

Mixed-In Trait

trait ShortLogger extends Logged {
  val maxLength = 15 ...
}
class Account {
  protected var balance = 0.0
}
class SavingsAccount extends Account with ShortLogger {var interest=0.0;...}

Super-class instance variables

SavingsAccount object

balance

Interest

maxLength

Sub-class instance variables
Extends Multiple Traits

- When a class extends multiple traits which define method of different names, there is no ambiguity about method calls.
- When a class extends multiple traits which define methods with the same name, calling the method name would invoke which of these methods?
- Answer: the method that is defined at the trait that is placed at the end of the extends.

Simulate Multiple Inheritance

```
abstract class Person
  parkingFee()
  itemDueDate()

trait Student
  parkingFee()
  itemDueDate()

trait Worker
  parkingFee()
  itemDueDate()

class StudentWorker extends Worker with Student

var alice=new WorkerStudent(12);
alice.parkingFee;?/?
alice.itemDueDate;?/

var bob=new StudentWorker(21);
bob.parkingFee;?/
bob.itemDueDate;?/
```

Assignment 8

- parkingFee
  Scala> val calendar = new java.util.GregorianCalendar();
  calendar: java.util.GregorianCalendar = scala>
  calendar.get (java.util.Calendar.DAY_OF_WEEK)
  res14: Int = 6
  scala> calendar.get (java.util.Calendar.HOUR_OF_DAY)
  res15: Int = 2

- itemDueDate
  Scala> val now=java.util.Calendar.getInstance();
  now: java.util.Calendar = java.util.GregorianCalendar
  scala> now.add(java.util.Calendar.DATE, +90);
  scala> now.getTime() res2: java.util.Date = Thu Feb 21 02:00:47 NST 2013