CSC 347 - Concepts of Programming Languages

Scala Classes

Instructor: James Riely

Learning Objectives

❓ Complex data in object-oriented programming?

- Identify classes and objects in Scala
- Use class parameters to initialize, query, and modify object values
- Use Scala objects to implement singletons
- Describe methods

Classes

- Define a class `C`

```
1 class C (f1:Int, val f2:Int, var f3:Int):
2   //...
```

- Has one constructor with parameters `f1`, `f2`, `f3`
- Instantiate the class `C`

```
1 val c = new C (2, 3, 5)
```

- Instance of `C` is heap allocated
- `c` is a reference to instance

Class Parameter Details

- Class parameters simultaneously declare constructors and accessor methods
- Components of the class body are `public` by default

Scala

```
1 class C1 (x:Int):
2   def double() = x+x
3 end C1
```

- `x` private immutable
- public constructor
- public `double`

Expressed in Java

```
1 public class C1 {
2   private final int x;
3
4   public C1(int x) { this.x = x; }
5
6   public int double() { return x+x; }
7 }
```

Class Parameter Details

- Immutable class parameters can be initialized but not modified

Scala

```
1 class C2 (val x:Int):
2   def double() = x+x
3 end C2
```

- `x` public immutable

Expressed in Java

```
1 public class C2 {
2   private final int x;
3
4   public C2(int x) { this.x = x; }
5
6   public int x() { return x; }
7
8   public int double() { return x+x; }
9 }
```

Class Parameter Details

- Mutable class parameters can be changed even after initialization

Scala

```
1 class C3 (var x:Int):
2   def double() = x+x
3 end C3
```

- `x` public mutable

Expressed in Java

```
 1 public class C3 {
 2   private int x;
 3
 4   public C3(int x) { this.x = x; }
 5
 6   public int x() { return x; }
 7
 8   public void setX(int x) { this.x = x; }
 9
10   public int double() { return x+x; }
11 }
```

Class Body

- Class body contains
  - Field declarations (`val` or `var`)
  - Constructor code
  - Method definitions

```
1  class C (f1:Int, val f2:Int, var f3:Int):
2    val f4 = f1 * f2
3    var f5 = f2 * f3
4
5    println ("Constructing instance of C")
6
7    def m (x:Int) : Int =
8      // cannot reassign to f1, f2, f4
9      f3 = f3 + 1
10     f5 = f5 + 1
11     f1 * f3 * x
12   end m
13 end C
```

Methods vs. Fields

- `val` field: `val x = 1`; strict, initialized once
- `def` non-parameterized method: `def y = 2`; non-strict, executed every time
- `lazy val`: memoized `def`, initialized on demand

Scala

```scala
class C:
  val x = 1
  def y = 2
  lazy val z = x + y
end C
```

Expressed in Java

```java
public class C {
  public final int x = 1;

  public int y() { return 2; }

  private Integer z = null;
  public int z() {
    if (z == null) z = x + y();
    return z;
  }
}
```

Objects

- `object` declares a single instance, accessible through the object name
- Language support for the singleton design pattern

```
1 object C:
2   var count = 0
3 end C
4
5 C.count = C.count + 1
```

Objects

- Singleton objects `object` are instantiated on program startup
- Method `main` must be declared in an `object`

Java

```java
public class C {
  public static void main (String[] args) {
    //...
  }
}
```

Scala

```scala
object C:
  def main (args:Array[String]) : Unit =
    //...
  end main
end C
```

Companion Objects

Java: static vs. non-static

- `static` components belong to a class
- All other belong to instance of class

```java
class C {
  private int f1 = 0;
  private static int f2 = 0;

  public int m1 () {
    C.f2 = 3;
    return f1;
  }

  public static int m2 (C other) {
    other.f1 = 5;
    return f2;
  }
}
```

Scala: Companion object replaces `static`

- All data and methods belong to objects
- Companion related to other instances

```scala
class C:
  private var f1 = 0
  def m1 () =
    C.f2 = 3
    f1
end C

object C:
  private var f2 = 0
  def m2 (other: C) =
    other.f1 = 5
    f2
end C
```

Coding Exercise

- `Matrix` class

Summary

- Class parameters and factory methods replace constructors
- Builtin support for the Singleton design pattern

Classes

```
1 class C
2  (     f: Int,
3   val g: Int,
4   var h: Int)
```

- `f` private immutable
- `g` public immutable
- `h` public mutable

Objects

- Replace `static`

```
1 object O:
2   val n = 10
3 end O
4
5 val n = O.n
```

Companion Objects

```
1 class X:
2   private val n = 10
3
4 object X:
5   def getN(x: X) = x.n
6
7 val n = X.getN(new X)
```

- Have access to `private` elements of companions