EDAA45 Programmering, grundkurs

Läsvecka 6: Klasser

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2016

6 Klasser

- Vad är en klass?
- Instansiering: `new`
- Referens saknas: `null`
- Synlighet
- Klasser i Java
- Getters och setters
- Implementation saknas: ???
- Klass-specifikationer
- Likhet
- Case-klasser
- Grumligt-lådan
- Veckans uppgifter

Vad är en klass?

Vad är en klass?

- En klass är en mall för att skapa objekt.
- Objekt skapas med **`new`** `Klassnamn` och kallas för **instanser** av klassen `Klassnamn`.
- En klass innehåller medlemmar (eng. *members*):
  - **attribut**, kallas även fält (eng. *field*): **`val`**, **`lazy val`**, **`var`**
  - **metoder**, kallas även operationer: **`def`**
- Varje instans har sin uppsättning värden på attributen (fälten).

Vad är en klass?

Metafor: En klass liknar en **stämpel**

- En stämpel kan tillverkas – motsvarar deklaration av klassen.
- Det händer inget förrän man stämplar – motsvarar `new`.
- Då skapas avbildningar – motsvarar instanser av klassen.

Syntax för klassdeklarationer

Klassens ”anatomi”

Exempel: Klassen Complex

Instansiering: new

Instansiering med new

Instansiering med fabriksmetod

Instansiering med default-argument

Instansiering med alternativa fabriksmetoder

Förändringsbar eller oföränderlig?

Referens saknas: `null`

Referens saknas: `null`

Konstruktor

Skräpsamling

Destruktor

Synlighet

Synlighet

definiera/förklara: private private[this]

Kompanjonsobjekt

Synlighet av klassparametrar i klasser & case-klasser

`private[this]` är ännu mer privat än `private`

```
class Hemlis(private val hemlis: Int) {
  def ärSammaSom(annan: Hemlis) = hemlis == annan.hemlis   // Funkar!
}

class Hemligare(private[this] val hemlis: Int) {
  def ärSammaSom(annan: Hemligare) = hemlis == annan.hemlis //KOMPILERINGSFEL
}
```

Vad händer om man inte skriver något? Olika för klass och case-klass:

```
class Hemligare(hemlis: Int) { // motsvarar private[this] val
  def ärSammaSom(annan: Hemligare) = hemlis == annan.hemlis //KOMPILERINGSFEL
}

case class InteHemlig(seMenInteRöra: Int) { // blir automatiskt val
  def ärSammaSom(annan: InteHemlig): Boolean =
    seMenInteRöra == annan.seMenInteRöra
}
```

Klasser i Java

Klasser i Java

Statiska medlemmar

Getters och setters

Getters och setters i Java

Getters och setters i Scala

Ändra attributrepresentation utan att påverka existerande kod

Complex som polära koordinater i Java med privat attribut
Complex som polära koordinater med publika attribut om man har enhetlig access

Implementation saknas: ???

Implementation saknas: ???

Klass-specifikationer

Specifikationer av klasser i Scala

- Specifikationer av klasser innehåller information som *den som ska implementera* klassen behöver veta.
- Specifikationer innehåller liknande information som dokumentationen av klassen (scaladoc), som beskriver vad *användaren* av klassen behöver veta.

Specification **Person**

```
/** Encapsulate immutable data about a Person: name and age. */
case class Person(name: String, age: Int = 0){
  /** Tests whether this Person is more than 17 years old. */
  def isAdult: Boolean = ???
}
```

- Specifikationer av Scala-klasser utgör i denna kurs ofullständig kod som kan kompileras utan fel.
- Saknade implementationer markeras med ???
- Kommentarer utgör krav på implementationen.

Specifikationer av klasser och objekt

Specification **MutablePerson**

```
/** Encapsulates mutable data about a person. */
class MutablePerson(initName: String, initAge: Int){
  /** The name of the person. */
  def getName: String = ???

  /** Update the name of the Person */
  def setName(name: String): Unit = ???

  /** The age of this person. */
  def getAge: Int = ???

  /** Update the age of this Person */
  def setAge(age: Int): Unit = ???

  /** Tests whether this Person is more than 17 years old. */
  def isAdult: Boolean = ???

  /** A string representation of this Person, e.g.: Person(Robin, 25) */
  override def toString: String = ???
}
object MutablePerson {
  /** Creates a new MutablePerson with default age. */
  def apply(name: String): MutablePerson = ???
}
```

Specifikationer av Java-klasser

- Specificerar signaturer för konstruktorer och metoder.
- Kommentarerna utgör krav på implementationen.
- Används flitigt på extentor i EDA016, EDA011, EDA017...
- Javaklass-specifikationerna behöver kompletteras med metodkroppar och klassrubriker innan de kan kompileras.

`class Person`

```
/** Skapar en person med namnet name och åldern age. */
Person(String name, int age);

/** Ger en sträng med denna persons namn. */
String getName();

/** Ändrar denna persons ålder. */
void setAge(int age);

/** Anger åldersgränsen för när man blir myndig. */
static int adultLimit = 18;
```

Likhet

Referenslikhet eller strukturlikhet?

- eq
- ==

Case-klasser

Case-klasser erbjuder strukturlikhet

Case-klass-godis:

- najs toString
- slipper skriva new
- == ger strukturlikhet

Grumligt-lådan

Grumligt-lådan

Veckans uppgifter

Övning: `classes`

- Kunna deklarera klasser med klassparametrar.
- Kunna skapa objekt med **`new`** och konstruktorargument.
- Förstå innebörden av referensvariabler och värdet **`null`**.
- Förstå innebörden av begreppen instans och referenslikhet.
- Kunna använda nyckelordet **`private`** för att styra synlighet i primärkonstruktor.
- Förstå i vilka sammanhang man kan ha nytta av en privat konstruktor.
- Kunna implementera en klass utifrån en specikation.
- Förstå skillnaden mellan referenslikhet och strukturlikhet.
- Känna till hur case-klasser hanterar likhet.
- Förstå nyttan med att möjliggöra framtida förändring av attributrepresentation.
- Känna till begreppen getters och setters.
- Känna till accessregler för kompanjonsobjekt.
- Känna till skillnaden mellan == och eq, samt != versus ne.

Laboration: `turtlegraphics`

- Kunna skapa egna klasser.
- Förstå skillnaden mellan klasser och objekt.
- Förstå skillnaden mellan muterbara och omuterbara objekt.
- Förstå hur ett objekt kan innehålla referenser till objekt av andra klasser, och varför detta kan vara användbart.
- Träna på att fatta beslut om vilka datatyper som bäst passar en viss tillämpning.