Functions and Data

Classes

- In Scala, we do this to define a class:

```
class Rational (x: Int, y: Int){
  def numerator = x
  def denominator = y
}
```

- This definition introduces two entities:
  - A new type named Rational
  - A constructor Rational to create elements of this type
- Scala keeps the names of types and values in different namespaces, so there is no conflict between the two entities named Rational

Objects

- We call the elements of a class type objects
- We create an object by calling the constructor of the class
- Objects of the class Rational have two members: ***numerator*** and ***denominator***
- We select the members of an object with the infix operator '.'

Methods

- One can go further and also package functions operating on a data abstraction in the data abstraction itself; such functions are called methods
- Here's a possible implementation for Rational methods:

```
class Rational(x: Int, y: Int){
  def numerator = x
  def denominator = y
  def add(r: Rational) =
      new Rational(numerator * r.denominator + r.numerator * denominator,
      denominator * r.denominator)
  override def toString =
      s"${numerator}/$denominator"
}
```

- **Note**: s"..." *in toString is an interpolated string, with values numerator and denominator in the places enclosed by ${...} (for more complex selectors) or by $ (for simple selections).*
- **Remark**: the modifier *override* declares that ***toString*** redefines a method that already exists (in the class java.lang.Object).

Exercise:

1. *Add a method* ***neg*** *to class Rational that is used like this:* ***x.neg -> evaluates to -x***
2. *Add a method sub to subtract two rational numbers.*

```
class Rational(x: Int, y: Int){
  def numerator = x
  def denominator = y
  def add(r: Rational) =
    new Rational(numerator * r.denominator + r.numerator * denominator,
 denominator * r.denominator)
  override def toString = s"${numerator}/$denominator"

  def neg = new Rational(-numerator, denominator)
  def sub(r: Rational) =
    this.add(r.neg)
 }
```