Tutorial
Object-Oriented Programming

Question 1.

Extend class Rational by adding:

a) new constructor without any parameter that returns a Rational(0,1)

b) new operator + with one parameter of type Int

c) new operator * with one parameter of type Rational

d) new operator * with one parameter of type Int

Question 2.

Create class UniformElement as a subclass of class Element such that an object of class UniformElement can be created by 3 parameters: a character c, number of rows and number of columns. And then add one more method *elem* in object Element to create this new kind of Element
For example, **Element.elem('c',2,3)** will create a UniformElement object that is equal to the ArrayElement object created by **Element.elem(Array("ccc","ccc"))**.

Question 3.

Write a new method called *checkEqual* that can check if two Element objects have the same content ? Think where the method should be declared.
For example,
val x = Element.elem(Array("ccc","ccc"))
val y = Element.elem('c',2,3)
x.checkEqual(y) $\Rightarrow$ true

Question 4.

Look at the example on Case class

a) make an object that represents the expression "(x + 1.2) * 4"

b) write method eval that can evaluate an expression without variable and return a **Number** object. The operators which may be appeared in an expression are "+", "-", "*", "/". For example,
val t = BinOp("*",Number(1.2),Number(2))
t.eval() $\Rightarrow$ Number(2.4)

Question 5.

Based on the example 2 on Traits, create a new trait such that a special queue can be easily created. Just the double of even integers can be put in the special queue. For example,
val queue = new BasicIntQueue with ...
queue.put(2) //4 is put in queue because 2 is even integer and 4 is double of 2
queue.put(3) //nothing is put in queue because 3 is an odd integer
queue.put(4) //8 is put in queue
queue.get() ⇒ 4
queue.get() ⇒ 8