UFAZ / Strasbourg University

Object Oriented Programming

Year 1 – Common curriculum

Tutorial / Lab session #4: Abstract classes and parametrized types

Exercise 1 – Basic arithmetic expressions (cont'd)

We can see that the code in the classes `Sum`, and `Product` (defined during the last lab) are quite similar: class members or constructors are identical. We can go further and make the implementation of `String asString()` and `float asValue()` identical as well:

1. What is the only difference in the method `String asString()` in your classes?
    a. In your classes, write a method `String label()` that returns this character.
    b. Re-write the method `String asString()` in your classes so they use the method written in the previous question.
    c. What can you say about the implementations of `toString()`?
2. Write the method `float eval(float arg1, float arg2)` in your classes. This method must return the following values:

	Class `Sum`	Class `Product`
`eval()`	arg1 + arg2	arg1 * arg2

   Now, re-write the method `asValue()` of your classes so they use the method `eval(float arg1, float arg2)`.
3. Create an abstract class `BinaryOperator` that will factorize the code of the methods `asValue()` and `asString()`. The classes `Sum` and `Product` must extend the class `BinaryOperator`.
    a. What are the abstract methods of this class?
    b. Should the class `BinaryOperator` implement the interface `ArithmeticExpression`? Why?

Exercise 2 – Parametrized arrays

Even though there are built-in parametrized collections in the Java API (such as the class `ArrayList<E>`), in this exercise we will implement a parametrized collection from scratch.

1. Create a class `MyArray<E>`. This class has 3 member variables: an array of Objects, a maximal capacity and a size (the current number of elements in the array). The capacity of the array is set when the object is instantiated (*i.e.* the capacity is given as parameter to the constructor).
2. Write a method `int size()` that returns the number of elements in the array.
3. Write a method `boolean isEmpty()` that returns true is the array is empty, else false.
4. Write a method `void add(E e)` that appends an element to the array. Make sure you are not trying to insert an element out of the bounds of the array!
5. Write a method `void remove(int index)` that removes the element at the index given as parameter. Make sure the index is valid. Don't forget to "shift" the other elements in the array so you don't end up with an array with "holes".
6. Write a method `E get(int i)` that returns the element at index i. Make sure the index is valid.
7. Override the method `String toString()` so you can print the array "properly".
8. Test your class in the `main` method