Exercise on JUnit
Exercise 1: shopping cart

- Exploit JUnit to test the following program
- http://didattica.agentgroup.unimore.it/wiki/images/6/65/ShoppingCart.zip

- Specifications
  - When created, the cart has 0 items
  - When empty, the cart has 0 items
  - When a new product is added, the number of items must be incremented
  - When a new product is added, the new balance must be the sum of the previous balance plus the cost of the new product
  - When an item is removed, the number of items must be decreased
  - When a product not in the cart is removed, a ProductNotFoundException must be thrown
    - Hint: insert the call in a try block and put a fail() after the call to removeItem()

- By Filippo Ricca DISI, Università di Genova, Italy
import java.util.*;

public class ShoppingCart {
    private ArrayList items;
    public ShoppingCart() {
        items = new ArrayList();
    }
    public double getBalance() {
        double balance = 0.00;
        for (Iterator i = items.iterator(); i.hasNext();){
            Product item = (Product)i.next();
            balance += item.getPrice();
        }
        return balance;
    }
}
public void addItem(Product item) {
    items.add(item);
}

public void removeItem(Product item) throws ProductNotFoundException {
    if (!items.remove(item)) {
        throw new ProductNotFoundException();
    }
}

public int getItemCount() {
    return items.size();
}

public void empty() {
    items.clear();
}

public class Product {
    private String title;
    private double price;
    public Product (String t, double p) {
        this.title = t;
        this.price = p;
    }
    public String getTitle() {
        return title;
    }
    public double getPrice() {
        return price;
    }
}
public double getPrice() {
    return price;
}

public boolean equals(Object o) {
    if (o instanceof Product) {
        Product p = (Product)o;
        return p.getTitle().equals(title);
    }
    return false;
}

return false;
}
Class ProductNotFoundException

```java
public class ProductNotFoundException extends Exception {
    public ProductNotFoundException() {
        super();
    }
}
```
Exercise 2: converter

- Write a class with a static method that converts a string into an integer value

Specifications
- The method must accept a string and convert it into an integer
- Well formed strings do not contain characters different from numbers, trailing spaces and minus
- The represented number must be in the range \([-32768, 32767]\)
- No real number are allowed

OK: “-3”, “500”, “-10”, “32767”
NO: “2 3”, “32768”, “A3”, “2.3”
Exercise 2: converter (2)

- Exploit JUnit to test the defined method
- Test also boundary conditions
- Hint 1: throw an exception in the converter method, and test if the exception has been thrown when the method is called with bad arguments
- Hint 2: exploit the `Integer.parseInt()` method for both the conversion and the check

By Filippo Ricca DISI, Università di Genova, Italy