# Problem 3. Easter Basket



*It’s Easter and we have a basket full of Easter eggs. We want the basket to look beautiful, so we have to do some changings.*

**Preparation**

Download the skeleton provided in Judge. **Do not** change the **packages**!

**Pay attention to name the package easterBasket, all the classes, their fields and methods the same way they are presented in the following document. It is also important to keep the project structure as described.**

**Problem description**

Your task is to create a repository which stores Easter eggs by creating the classes described below.

First, write a **JAVA** class **Egg** with the following properties:

* **color: String**
* **strength: int**
* **shape: String**

The class **constructor** should receive **color, strength and shape**. You need to create the appropriate **getters and setters**. Override the **toString()** method in the following format:  
**"{color} egg, with {strength} strength and {shape} shape."**

**Next**, write a **JAVA** class **Basket** that has **data** (a List which stores the entity **Egg**). All entities inside the repository have the **same properties**. Also, the **Basket** class should have those **properties**:

* **material: String**
* **capacity: int**

The class **constructor** should receive **material** and **capacity**, also it should initialize the **data** with a new instance of the collection.Implement the following features:

* **List<Egg> data** - **collection** that holds added eggs
* **Method addEgg(Egg egg)** – **adds** an **entity** to the data **if** **there** **is** **room** for it
* **Method removeEgg(string color)** – removes an egg by **given color,** if such **exists**, and **returns boolean** (true if it is removed, otherwise – false)
* **Method getStrongestEgg()**– **returns the strongest egg**
* **Method getEgg(string color)** – **returns** the **egg** with the **given color**
* **Method getCount** – **returns** the **number** of **eggs**
* **Method report()** – **returns** a **string** in the following **format** (print the eggs in **order of appearance**):
  + **"{material} basket contains:  
    {Egg1}  
    {Egg2}  
    (…)"**

**Constraints**

* The **color** and **strength** of the eggs will be **always unique**.
* You will always have an egg added before receiving methods manipulating the Basket’s eggs.

**Examples**

This is an example how the **Basket** class is **intended to be used**.

|  |
| --- |
| **Sample code usage** |
| *// Initialize the repository (Basket)* Basket basket = **new** Basket(**"Wood"**, 20); *// Initialize entity (Egg)* Egg egg = **new** Egg(**"Red"**, 10, **"oval"**); *// Print Egg* System.***out***.println(egg); *// Red egg, 10 strength, oval shape  // add Egg* basket.addEgg(egg);  *// remove Egg* System.***out***.println(basket.removeEgg(**"Pink"**)); *//False* Egg secondEgg = **new** Egg(**"Green"**, 9, **"pointy"**);  *// add Egg* basket.addEgg(secondEgg);  *// get strongest egg* Egg strongestEgg = basket.getStrongestEgg();  System.***out***.println(strongestEgg);  *// get egg* Egg getEgg = basket.getEgg(**"Green"**); *// Green egg with 9 strength, pointy shape* System.***out***.println(getEgg);   System.***out***.println(basket.report()); *// Wood basket contains: // Red egg, with 10 strength and oval shape. // Green egg, with 9 strength and pointy shape.* |

**Submission**

Submit **single .zip file**, containing " **easterBasket**" package, **with the classes inside** (**Basket** and **Egg**) and the **Main** **class**, there is no specific content required inside the **Main** class e. g. you can do any kind of local testing of you program there. However, there should be **main(String[] args)** method inside.