Introduction

Storing data in static properties is most often not a good idea. In task list 8 (which focused on routing rules), the game only appears to be working properly. It is worth testing it by running at least two different web browsers (simulating the operation of two application users). It will quickly turn out that a draw made in one browser affects the game in the other browser. The reason is precisely the use of static elements in the controller. To ensure that each user of your application plays an independent game, you should use **session variables** or **cookies.**

Tasks list

The task does not explicitly present the classes that should be created to handle the basket of goods, but they should appear in the code.

1. Modify the guessing game from list 8 so that each user plays independently of the other. Use **session variables**.
2. To solve the task from list 10 for `ShopController`, add remembering the shopping cart **in cookies**. The page showing the goods (list 10, task 6) should have a button/link for each product to add the product to the cart. If the user selects the same product several times, it will be added in several pieces (i.e. remember the product **ID** (in the key) and its **amount** in the basket (in the value) **in the cookies**, (e.g. if in the basket there are 2 articles of the ID=15, the key in cookie will be "**article15**", and the value 2). Each user (e.g. you can test by using two different browsers) should have their own, independent basket. The contents of the basket should be remembered for a week, even if the user closes the browser
3. The store should have a **button**/link to the cart **in the menu**. Using it, you can view the contents of the basket on a new page (a view similar to the shop view, but with the number of items in the basket). On this new page you can also increase or decrease the number of pieces of a given product, or remove it completely from the basket. After each such change, the website continuously calculates the total cost of the goods in the basket. If there is nothing in the cart, the website should visibly inform the user about this.
    a. Basic version: in the basket view, adding/removing an item triggers a GET query to the appropriate controller action, which will modify the contents of the basket and show its new contents.
    b. JS Issue: Is it possible to do this just locally within the browser using JavaScript code without forcing a new GET query? Will the data in the browser and on the server be consistent?

Question: How do operations on the basket relate to the operation of **deleting** an item/category, which another user of your application can perform **in the meantime**?

Deadline I: Meeting 12 (100 points)
Deadline II: Meeting 13 (80 points)
Deadline III: Meeting 14 (50 points)