# 03. Vending Machine



*You are tasked with creating a program that simulates a vending machine for hot drinks. The vending machine should be able to store and dispense various types of hot drinks. You should create the necessary classes and methods to implement this functionality.*

**Preparation**

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

**Pay attention to name the package vendingSystem, 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 that stores drinks by creating the classes described below.

### Drink

You are given a class **Drink** with the following properties:

* **name - String**
* **price - BigDecimal**
* **volume - int**

The class **constructor** should receive **name, price** and **volume**.

Override the **toString()** method in the following format:  
**"Name: {name}, Price: ${price}, Volume: {volume} ml"**

### VendingMachine

**Next**, you are given a class **VendingMachine** that has **Drinks** (a List that stores Drinks). All entities inside the repository have the **same properties**. The **VendingMachine** class should have the following **properties**:

* **buttonCapacity - int**
* **drinks – List<Drink>**

The class **constructor** should receive **buttonCapacity**, also it should initialize the **Drinks** with a new instance of the collection.Implement the following features:

* **getCount() - int** - **returns** the number of **drinks**, **available** in the Vending machine.
* **Method addDrink(Drink drink)** – **adds** an **entity** to the **collection** of Drinks, **if** the **Capacity** **allows it**.
* **Method removeDrink(String name)** – **removes** a **drink by** **given name,** if such **exists**, and **returns boolean** (**true** if it is removed, otherwise – **false**)
* **Method getLongest()** – **returns** the **Drink** with the **biggest value** of **volume** property**.**
* **Method getCheapest() –** **returns** the **Drink** with the **lowest value** of **price** property.
* **Method buyDrink(String name) - returns a string** in the format of the overridden **toString()** method**.**
* **Method report()** – **returns** a **string** in the following **format**:
  + **"Drinks available:  
    {Drink1}  
    {Drink2}  
    (…)"**

### Constraints

* The **names** of the drinks will always be **unique**.
* You will always have drinks added before receiving methods, manipulating the drinks in the VendingMachine.

### Examples

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

|  |
| --- |
| **Sample code usage** |
| *//Initialize the repository (VendingMachine)* VendingMachine vendingMachine = new VendingMachine(6);  *//Initialize entity (Drink)* Drink tea = new Drink("Tea", new BigDecimal("1.5"), 300);  Drink coffee = new Drink("Coffee", new BigDecimal("2.0"), 120);  Drink hotChocolate = new Drink("Hot Chocolate", new BigDecimal("2.5"), 200);  Drink latte = new Drink("Latte", new BigDecimal("3.5"), 220);  Drink cappuccino = new Drink("Cappuccino", new BigDecimal("2.8"), 180);  Drink mocha = new Drink("Mocha", new BigDecimal("2.1"), 150);  Drink herbalTea = new Drink("Herbal Tea", new BigDecimal("1.75"), 300);  *//Get Count* System.*out*.println(vendingMachine.getCount()); *//0  //Add Drinks* vendingMachine.addDrink(tea);  vendingMachine.addDrink(coffee);  vendingMachine.addDrink(hotChocolate);  vendingMachine.addDrink(latte);  vendingMachine.addDrink(cappuccino);  vendingMachine.addDrink(mocha);  *//Try to add drinks when the capacity is full* vendingMachine.addDrink(herbalTea);  *//Get Count* System.*out*.println(vendingMachine.getCount()); *//6  //Remove Drink* System.*out*.println(vendingMachine.removeDrink("Herbal Tea")); *// False* System.*out*.println(vendingMachine.removeDrink("Tea")); *// True  //Get Longest Drink* System.*out*.println(vendingMachine.getLongest()); *//Name: Latte, Price: $3.5, Volume: 220 ml  //Get Cheapest Drink* System.*out*.println(vendingMachine.getCheapest()); *//Name: Coffee, Price: $2.0, Volume: 120 ml  //Buy a specific Drink* System.*out*.println(vendingMachine.buyDrink("Latte")); *//Name: Latte, Price: $3.5, Volume: 220 ml  //Drinks Report* System.*out*.println(vendingMachine.report());  *//Drinks available: //Name: Coffee, Price: $2.0, Volume: 120 ml //Name: Hot Chocolate, Price: $2.5, Volume: 200 ml //Name: Latte, Price: $3.5, Volume: 220 ml //Name: Cappuccino, Price: $2.8, Volume: 180 ml //Name: Mocha, Price: $2.1, Volume: 150 ml* |

**Submission**

Zip all the files in the project folder except the **bin** and **obj** folders

Submit a **single .zip file**, containing **vendingSystem package, with the classes inside (VendingMachine, Drink, 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 your program there. However, a **main(String[] args)** method should be inside.