Classes -Exercises

[1. Car Info 1](#_Toc145518976)

[2. Car Constructors 2](#_Toc145518977)

[3. Bank Account 3](#_Toc145518978)

[4. Class Vehicle 4](#_Toc145518979)

[5. Class Storage 4](#_Toc145518980)

[6. Randomize Words 5](#_Toc145518981)

[7. Students 6](#_Toc145518982)

[8. Articles 7](#_Toc145518983)

# Car Info

Create a class named **Car**.

The class should have **public** fields for:

* Brand: **String**
* Model: **String**
* Horsepower: **int**

Create a **new class** and ensure **proper naming.**

Define **private Fields.**

Change the access modifiers of all class fields to **private.**

**Create getters** and **setters** for each class field.

**Create Car Info Method**

This method should return the info about any car object in the following format:

"**The car is: {brand} {model} – {horsePower} HP.**"

You must figure out how to create a method and use it in the outside code:  
System.out.println(car.carInfo())

**Test the Program**

Read cars objects, add them to the collection of your choice, and print each one on the console using the **carInfo()** method. The input consists of a single integer **N,** the number of lines representing car objects. On each line you will read car info in the following format "**{brand} {model} {horsePower}**"separated by single space.

**Examples**

|  |  |
| --- | --- |
| Input | Output |
| 3  Toyota Rav4 390  Mercedes Benz 500  Volga 24 49 | Info: Toyota Rav4 - 390 HP.  Info: Mercedes Benz - 500 HP.  Info: Volga 24 - 49 HP. |
| 5  Toyota Corola 1  Toyota Rav4 2  BMW Q5 3  BMW Q12 4  BeEmVe Z1 5 | Info: Toyota Corola - 1 HP.  Info: Toyota Rav4 - 2 HP.  Info: BMW Q5 - 3 HP.  Info: BMW Q12 - 4 HP.  Info: BeEmVe Z1 - 5 HP. |

# Car Constructors

Make proper constructors for the Car class so you can create car objects with a different type of input information.

If you miss information about the field of **type String** set the value to "**unknown**",and for an **integer => -1.**

First, **declare** a **constructor** which takes only the car brand as a parameter and a **constructor** which **sets** all the **fields**.

Read information about cars the same way as the previous task, however, this time use **constructors** to create the objects. You should be able to handle **different** **types** of input, the format will be the same as the previous task, but this time some of the data may be missing. For example, you can get only the car **brand** – which means you have to set the car model to "**unknown**"and the Horsepower value to **-1**. There will be lines with **complete** car data, declare constructor which sets all the fields.

Add the car objects to a **collection** of your choice and print the data as in the previous task. The input will **always** have one or three elements on each line.

**Examples**

|  |  |
| --- | --- |
| Input | Output |
| 2  Chevrolet  Golf Polo 49 | The car is: Chevrolet unknown - -1 HP.  The car is: Golf Polo - 49 HP. |
| 2  Toyota  Toyota Rav4 | The car is: Toyota unknown - -1 HP.  The car is: Toyota Rav4 - -1 HP. |

# Bank Account

Create class **BankAccount**.

The class should have **private fields** for:

* Id: **int** (Starts from **1** and **increments** for every **new** **account**)
* Balance: **double**
* Interest rate: **double** (Shared for all accounts. **Default value: 0.02**)

The class should also have **public** methods for:

* **setInterestRate(double interest):** **void (static)**
* **getInterest(int Years):** **double**
* **deposit(double amount):** **void**

Create a test client supporting the following commands:

* **Create**
* **Deposit {Id} {Amount}**
* **SetInterest {Interest}**
* **GetInterest {ID} {Years}**
* **End**

**Examples**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comments |
| Create  Deposit 1 20  GetInterest 1 10  End | Account ID1 created  Deposited 20 to ID1  4.00 |  |
| Create  Create  Deposit 1 20  Deposit 3 20  Deposit 2 10  SetInterest 1.5  GetInterest 1 1  GetInterest 2 1  GetInterest 3 1  End | Account ID1 created  Account ID2 created  Deposited 20 to ID1  Account does not exist  Deposited 10 to ID2  30.00  15.00  Account does not exist | Sets the global interest rate to 1.  Prints interest for a bank account with id 1 for 1 year period. |

# Class Vehicle

Create a class with name **Vehicle** that has the following properties:

**type** – a string

**model** – a string

**engine** – an object that contains:

-**power** – number

**fuel** – a number

**drive** – a method that receives fuel loss and decreases the fuel of the vehicle by that number.

The **constructor** should receive the **type**, the **model**, the **engine** and the **fuel**

**Example**

Test your Vehicle class

|  |  |
| --- | --- |
| **Input** | Output |
| Engine - power: 100  var vehicle = new Vehicle('a', 'b', engine, 200);  vehicle.drive(100);  print(vehicle.fuel) | 100 |

# Class Storage

Create a class **named Storage**. It should have the following **properties**:

**capacity** – a number that **decreases when adding a given quantity** of products in storage

**storage** – **list of products** (object). **Each product** should have:

**name** - a string

**price** – a number (price is for a single piece of product)

**quantity** – a number

**totalCost** – sum of the cost of the products

**addProduct** – a function that receives a product and adds it to the storage

**getProcuts** – a function that returns all the products in storage in JSON format, each on a new line

The **constructor** should receive a **capacity**

Test your Storage class

|  |  |
| --- | --- |
| **Input** | Output |
| productOne = {name: cucumber, price: 1.50, quantity: 15}  productTwo = {name: 'tomato', price: 0.90, quantity: 25}  productThree = {name: 'bread', price: 1.10, quantity: 8}  var storage = new Storage(50)  storage.addProduct(productOne)  storage.addProduct(productTwo)  storage.addProduct(productThree)  storage.getProducts()  print(storage.capacity)  print(storage.totalCost) | 2  53.8 |

# Randomize Words

You are given a **list of words in one line**. **Randomize their order** and print each word on a separate line.

**Examples**

|  |  |  |
| --- | --- | --- |
| Input | Output | Comments |
| Just have fun with Java | Java  Just  fun  have  with | The order of the words in the output will be different after each program execution. |
| Java is one of the programming languages | the  programming  best  one  languages  is  of  Java |  |

**Hints**

* **Split** the input string (by space) and create an **array of words.**
* Create a random number generator - an object **rnd** of type **Random.**
* In a **for-loop exchange, each number** at positions 0, 1, …, **words.Length-1** by a number at **random. position**. To generate a random number in rangeuse **rnd.nextInt(words.length)**.
* Print each word in the array on a new line.

# Students

Define a class **Student**, which holds the following information about students: first name, last name, age, and hometown.

Read the list of students until you receive the "**end**" command. After that, you will receive a city name. Print only students which are from the given city, in the following format: **"{firstName} {lastName} is {age} years old"**.

**Examples**

|  |  |
| --- | --- |
| Input | Output |
| John Doe 15 Sofia  Peter Peterov 14 Plovdiv  Linda Bridge 16 Sofia  Simeon Bond 12 Varna  end  Sofia | John Doe is 15 years old  Linda Bridge is 16 years old |
| Anthony Taylor 15 Chicago  David Anderson 16 Washington  Jack Lewis 14 Chicago  David Lee 14 Chicago  end  Chicago | Anthony Taylor is 15 years old  Jack Lewis is 14 years old  David Lee is 14 years old |

# Articles

Create an article class with the following properties:

* **Title** – a string
* **Content** – a string
* **Author** – a string

The class should have a constructor and the following methods:

* **Edit (new content)** – change the old content with the new one
* **ChangeAuthor (new author)** – change the author
* **Rename (new title)** – change the title of the article
* override **ToString** – print the article in the following format:

**"{title} - {content}: {author}"**

Write a program that reads an article in the following format **"{title}, {content}, {author}"**. On the next line, you will get the number **n**. On the next **n lines,** you will get one of the following commands:

* **"Edit: {new content}"**
* **"ChangeAuthor: {new author}"**
* **"Rename: {new title}"**.

**Examples**

|  |  |
| --- | --- |
| Input | Output |
| some title, some content, some author  3  Edit: best content  ChangeAuthor: best author  Rename: best title | best title - best content: best author |
| Foundation, Brilliant, Isaak Asimov  3  ChangeAuthor: Tolkien  ChangeAuthor: Martin  ChangeAuthor: Rowling | Foundation - Brilliant: Rowling |