Java module 3

Exercises Day 2 (B)

1 - Enum	Toy shop
Instructions	We will create a program to manage a toy shop inventory. Step 1: Create an Enum called ToyType to allow classifying toys into categories. Some of the categories could be: STUFFED_ANIMAL, VEHICLE or PUZZLE. Step 2: Create a class to represent a toy that exists within the toy shop. It should be possible to store the name of the toy and its type. Step 3: Create a main program that will allow the store manager to add toys to the shop's inventory. Step 4: Run the program and take a look at the objects that were created using the debugger.

Expected output	Toy name:
	>>> HotWheels Winter car
	Toy type (STUFFED_ANIMAL, VEHICLE, PUZZLE):
	>>> VEHICLE
	Toy added to the inventory. Would you like to add another toy (y/n)?
	>>> y
	Toy name:
	>>> Brown teddy
	Toy type (STUFFED_ANIMAL, VEHICLE, PUZZLE):
	>>> STUFFED_ANIMAL
	Toy added to the inventory. Would you like to add another toy (y/n)?
	>>> n
	Good bye!

Solution ToyType.java	(see code below)

```java
public enum ToyType {
    STUFFED_ANIMAL, PUZZLE, VEHICLE;

    /**
     * Function to get the list of toy types as a string
     */
    public static String getToyTypes() {
        String toyTypesString = "";
        ToyType[] toyTypes = ToyType.values();
        for(int i = 0; i < toyTypes.length; i++) {
            toyTypesString += toyTypes[i];
            if (i < toyTypes.length - 1) {
                toyTypesString += ", ";
```

| | ```java
            }
        }
        return toyTypesString;
    }
}
``` |
| Solution<br>Toy.java | ```java
public class Toy {
    private String name;
    private ToyType toyType;

    public Toy(String name, ToyType toyType) {
        this.name = name;
        this.toyType = toyType;
    }

    public Toy(String name, String toyType) {
        this.name = name;
        this.toyType = ToyType.valueOf(toyType);
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public ToyType getToyType() {
        return toyType;
    }

    public void setToyType(ToyType toyType) {
        this.toyType = toyType;
    }
}
``` |
| ToyStore.java | ```java
import java.util.Scanner;
import java.util.List;
import java.util.ArrayList;
``` |

```java
public class ToyStore {
    public static void main(String[] args) {

        //List to store the toys
        List<Toy> toys = new ArrayList<>();

        Scanner scanner = new Scanner(System.in);

        boolean exit = false;
        while (!exit) {
            System.out.print("Toy name: ");
            String toyName = scanner.nextLine();

            System.out.print("Toy type (" +
ToyType.getToyTypes() + "): ");
            String toyType = scanner.nextLine();

            toys.add(new Toy(toyName, toyType));
            System.out.println("Toy added to the inventory.
Would you like to add another toy (y/n)? ");
            String choice = scanner.nextLine();
            if (choice.equals("n")) {
                exit = true;
            }
        }

        scanner.close();
    }
}
```

2 - Inheritance	Adding books to the shop
Instructions	We will continue developing the program for the toy shop. The owner now wants to sell books.
	Step 1: Create a parent class called Article. The Article class should allow storing the price of the article, a unique code for the article and the amount of items in stock.
	Step 2: Create a class called Book that is a child class of Article. The Book class should allow storing the book's title and author.
	Step 3: Update the Toy class created in the previous exercise to be a

	child class of Article.
	Step 4: Modify the main program to allow adding Books and Toys to the inventory. Which would be the ideal structure to store the articles such that they can be searched by the article code later on?
	Step 5: Add a method to the Article class that will allow printing the article information to the output. Override this method for each of the child classes such that it prints all relevant information of each article.
	Step 6: Add a method to the Article class that will reduce the amount of items stored in stock. Modify the main program to allow the manager to remove items from stock once they are sold.

Expected output	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> t
	Toy name:
	>>> HotWheels Winter car
	Toy type (STUFFED_ANIMAL, VEHICLE, PUZZLE):
	>>> VEHICLE
	Toy price:
	>>> 3,5
	Amount of items in stock:
	>>> 15
	Toy added to the inventory.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> b
	Book title:
	>>> Harry Potter
	Book author:
	>>> J. K. Rowling
	Book price:
	>>> 27
	Amount of items in stock:
	>>> 100
	Book added to the inventory.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> p
	Toy (code 1): HotWheels Winter car - 3,5 euros - 15 items in stock.
	Book (code 2): Harry Potter by J. K. Rowling - 27 euros - 100 items in stock.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> r
	Enter the article code:
	>>> 2
	Stock updated.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> p
	Toy (code 1): HotWheels Winter car - 3,5 euros - 15 items in stock.

	Book (code 2): Harry Potter by J. K. Rowling - 27 euros - 99 items in stock.

3 - Polymorphism	Promotional price
Instructions	We will continue developing the program for the toy shop. Step 1: Create a polymorphic method to print the information of each article. This new method should have the same name as the one created in the previous exercise but it should receive the amount of discount that should be applied to the price of the articles. Step 2: Update the main program to allow the manager to print all articles with a certain discount.

Expected output	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove articles from stock?
	>>> t
	Toy name:
	>>> HotWheels Winter car
	Toy type (STUFFED_ANIMAL, VEHICLE, PUZZLE):
	>>> VEHICLE
	Toy price:
	>>> 3,5
	Amount of items in stock:
	>>> 15
	Toy added to the inventory.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove articles from stock?
	>>> b
	Book title:
	>>> Harry Potter
	Book author:
	>>> J. K. Rowling
	Book price:
	>>> 27
	Amount of items in stock:
	>>> 100
	Book added to the inventory.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove articles from stock?
	>>> p
	Would you like to apply a discount to the prices (y/n)?
	>>> n
	Toy (code 1): HotWheels Winter car - 3,5 euros - 15 items in stock.
	Book (code 2): Harry Potter by J. K. Rowling - 27 euros - 100 items in stock.
	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> r
	Enter the article code:
	>>> 2
	Stock updated.

	Would you like to (t) add a toy, (b) add a book (p) print the list of all articles or (r) remove an article from stock?
	>>> p
	Would you like to apply a discount to the prices (y/n)?
	>>> y
	Enter the discount as a percentage (between 1 and 99):
	>>> 20
	Toy (code 1): HotWheels Winter car - 2,8 euros - 15 items in stock.
	Book (code 2): Harry Potter by J. K. Rowling - 21,6 euros - 99 items in stock.