



## Homework 4. Abstract Classes, Wrapper Classes, Strings

Registration Deadline: 04.05.2025, 23:59

Hand-in Deadline: 07.05.2025, 23:59

### Exercise 1. Discount Drama

Create:

- `abstract class Article` with `int` attributes `discountThreshold` and `householdLimit`. It also defines two abstract methods:
  - `int getBulkDiscount()`
  - `boolean showWarning()`
- Example: If `discountThreshold = 5` for the article "salt", you receive a bulk discount when buying at least 5 packs of salt. If `householdLimit = 7` for salt, then buying 8 or more packs exceeds the household-appropriate quantity.
- `abstract class GroceryItem` extends `Article` with a `boolean` attribute `isPopular`.
- `abstract class PopularGrocery` extends `GroceryItem` with `int` attributes `popularityLevel` and `quantity`. It includes:
  - A constructor `PopularGrocery(int popularityLevel)` which sets `isPopular = true` and sets `this.popularityLevel`.
  - Implementation of `showWarning()`, which returns `true` if `quantity > householdLimit`.
  - Setter for `quantity`.
- `class Milk` extends `PopularGrocery`. It sets:
  - `this.discountThreshold = discountThreshold`
  - `this.quantity = quantity`
  - `this.householdLimit = 20`
- It implements `getBulkDiscount()` to return 12 if `quantity >= discountThreshold`, otherwise 0.
- `class Flour` extends `PopularGrocery`. It sets:
  - It has a constructor `Flour(int quantity, int discountThreshold, int popularityLevel)`.

- `this.quantity = quantity`
- `this.discountThreshold = discountThreshold`
- `this.householdLimit = 15`

It implements `getBulkDiscount()` to return 5 if `quantity >= discountThreshold`, otherwise 0.

## Exercise 2. Letter Soup Deluxe

Write a program that checks whether two strings are **anagrams** of each other — that is, they must contain the same letters in any order.

- In your `main` method:
  - Use the `Scanner` class to read two strings from the user.
  - Process the strings to ignore capitalization and spaces before checking for anagrams.
- Write a method `boolean isAnagram(String a, String b)` that returns `true` if the two strings are anagrams.
  - Implement a helper method `int[] countLetters(String s)` which counts how many times each letter (a–z) appears in the string.
  - Use two such arrays to compare the letter frequencies of `a` and `b`.
- Letter count output:  
Write a method `void printLetterCounts(String s)` that prints only the letters used in the input string, one per line, in the following format:  

```
a: 2
e: 1
r: 3
```

You may use the following as a starting point:

```
import java.util.Scanner;

public class AnagramChecker {

    /**
     * Returns true if the two strings are anagrams of each other.
     */
    public static boolean isAnagram(String a, String b) {
        // TODO: implement using countLetters()
        return false;
    }

    /**
     * Helper method to count how many times each letter (a-z) appears.
     */
    public static int[] countLetters(String s) {
        int[] counts = new int[26];
        // TODO: fill array with letter frequencies
    }
}
```

```

        return counts;
    }

    /**
     * Print only letters that appear in the string with their counts.
     */
    public static void printLetterCounts(String s) {
        // TODO: implement nicely formatted output
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        // Get two strings from the user
        System.out.print("Enter the first string: ");
        String word1 = scanner.nextLine();

        System.out.print("Enter the second string: ");
        String word2 = scanner.nextLine();

        // TODO: Process the strings here (ignore spaces, case)

        System.out.println("Are they anagrams? " + isAnagram(word1, word2));
        printLetterCounts(word1);
    }
}

```

- Comparing Strings: Explain the difference between == and equals() when comparing String objects in Java. Provide a short example showing why == may not be reliable.
- Consider the following code snippet and explain why this comparison returns true.
- What happens if you create the second string using new String("abc") instead?

```

String string1 = "abc";
String string2 = "abc";
System.out.println(string1 == string2); // true

```

### Exercise 3. Smart Switch

Design a tiny smart-home device exercise focusing on Java **interfaces**.

- **Create interface Switchable**
  - void turnOn(), void turnOff(), boolean isOn()
- **Create interface Dimmable**
  - Constant int MAX\_BRIGHTNESS = 100
  - void setBrightness(int level) (ensure the brightness level stays between 0 and MAX\_BRIGHTNESS)

- `int getBrightness()`
- Default method `void dimToHalf()` that sets brightness to `MAX_BRIGHTNESS / 2`
- **class** `Lamp` implements both interfaces with fields `boolean on` and `int brightness`.
  - `setBrightness` only changes the value when the lamp is on.
- **Demo:** In main create a lamp, turn it on, set brightness to 80, call `dimToHalf()`, then print the brightness.

You may start with:

```
public interface Switchable {
    // TODO: implement
}

public interface Dimmable {
    // TODO: implement
}

public class Lamp implements Switchable, Dimmable {

    @Override
    public void turnOn() {
        // TODO: implement
    }

    @Override
    public void turnOff() {
        // TODO: implement
    }

    @Override
    public boolean isOn() {
        // TODO: implement
        return false;
    }

    @Override
    public void setBrightness(int level) {
        // TODO: implement
    }

    @Override
    public int getBrightness() {
        // TODO: implement
        return 0;
    }

    public void dimToHalf() {
        // TODO: implement
    }
}
```

```
    }  
}  
  
public class LampDemo {  
  
    public static void main(String[] args) {  
        Lamp lamp = new Lamp();  
  
        // turn on, set brightness, dim to half, print result  
        lamp.turnOn();  
        lamp.setBrightness(80);  
        lamp.dimToHalf();  
        System.out.println("Brightness: " + lamp.getBrightness());  
    }  
}
```