Assignment 4

Refer to Canvas for assignment due dates for your section.

Objectives:
- Implement and test a mutable ADT.
- Implement and test an immutable ADT.

General Requirements

Create a new Gradle project for this assignment in your course GitHub repo. Make sure to follow the instructions provided in "Using Gradle with IntelliJ" on Canvas.

Create a separate package for each problem in the assignment. Create all your files in the appropriate package.

To submit your work, push it to GitHub and create a release. Refer to the instructions on Canvas.

Your repository should contain:
- One .java file per Java class.
- One .java file per Java test class.
- One pdf or image file for each UML Class Diagram that you create. UML diagrams can be generated using IntelliJ or hand-drawn.
- All non-test classes and non-test methods must have valid Javadoc.

Your repository should **not** contain:
- Any .class files.
- Any .html files.
- Any IntelliJ specific files.

Problem 1

Implement an ADT called `CookieCatalog` - a collection (catalog) which will be used by a cookie factory, to keep track of information about produced cookies and their nutritional information.

Class `Cookie` which stores relevant nutritional information, as well as information about ingredients, has already been developed for you (please download it from the Canvas page for this assignment).

The ADT `CookieCatalog` will need to support the following functionality.

- Method `isEmpty`() - checks if the `CookieCatalog` is empty.

- Method `countCookies`() - counts the number of `Cookiess` in the `CookieCatalog`.
- Method `checkCookie`(String cookieName) - checks if a `Cookie` with the requested name exists in the `CookieCatalog`.
- Method `addCookie(Cookie cookie)` - adds a `Cookie` to the `CookieCatalog`. Please note that `CookieCatalog` does not allow duplicate `Cookies`.
- Method `removeCookie(Cookie cookie)` – removes a specified `Cookie` from the `CookieCatalog`.If the `Cookie` does not exist in the `CookieCatalog`, the system should throw `CookieNotFoundException`, which you will have to implement yourself.
- Method `findGlutenFreeSugarLiteCookies()` - finds and return all `Cookies` from the `CookieCatalog` that are gluten-free, and whose amount of sugar per serving is less than 30 grams.
- Method `findNutsAndDairyCookies()` - finds and return all `Cookies` from the `CookieCatalog` that contain both nuts and dairy.

Specify this ADT in an interface and implement it, and any other classes needed to satisfy the specification. You should also implement `toString`, `equals`, and `hashCode` for this ADT.

You may not use any built-in Java collections, as the underlying data structure, but you are welcome to use either an array of your own implementation of class `Node` to implement `CookieCatalog` as a linked data collection.
Please do not modify the provided `Cookie` class.

Problem 2

In this assignment, your task is to implement a **polynomial as a linked collection of terms.**

A **polynomial** typically consists of several terms, and each term has a **coefficient (weight)**, and a **variable** raised to a **power**.

Polynomials are considered **one-variable polynomials** if all their terms contain only one variable. An example of a one-variable polynomial is:

$$f(x) = 3x^4 - 5x^3 + 2x - 4$$

This polynomial has four terms.

The **degree of a polynomial** is defined as the highest power of the variable in a term. In the above example, the degree of the polynomial is 4.

The polynomials we deal with will have only positive, integer powers. The coefficients of our polynomials will also be integers.

In this assignment we will be interested in the following two operations with polynomials:

- **Comparison between polynomials:** two polynomials are the same if they contain the same terms.
- **Polynomial addition:** two polynomials can be added together to create a new polynomial. The addition is performed by combining the terms, and adding the coefficients (weights) of the terms with the same power.
  For example, two polynomials *f*(*x*) and *g*(*x*):
  $$f(x)= 3x^4 - 5x^3 + 2x - 4 \text{ and } g(x) = 2x^3 + 2x^2 + 4$$
  can be added together as follows:
  $$f(x) + g(x) = 3x^4 - 5x^3 + 2x - 4 + 2x^3 + 2x^2 + 4 =$$
  $$= 3x^4 + (-5 + 2)x^3 + 2x^2 + 2x + (4 - 4)$$
  $$= 3x^4 = 3x^3 + 2x^2 + 2x$$

  The degree of the sum is the maximum of the degrees of the two polynomials.

The ADT `Polynomial` will need to support the following functionality:

- A method `getDegree`, that returns the degree of a polynomial.
- A method `getCoefficient`, that takes a power, and returns the coefficient for the term with that power.
- A method `addTerm`, that takes a coefficient and a power (both integer numbers), and adds the given term to an existing polynomial.
- A method `removeTerm`, that takes a power, and removes a term in the polynomial with that power.
- A method `isSame`, that takes another `Polynomial` object and returns `True` if they are the same, `False` otherwise.
- A method `add`, that takes another `Polynomial` object, and returns the resulting polynomial, obtained by adding the two polynomials.

Specify this ADT in an interface and **implement it as a linked collection of terms**. You are welcome to implement any other classes needed to satisfy the specification. You should also implement `toString`, `equals`, and `hashCode` for this ADT.

Additionally, implement method `printPolynomial(),` that returns a string representation of your polynomial.
The following examples should help you infer the required format:

$5x^2 + 4x - 2$ creates the string 5x^2 + 4x^1 - 2

$-50x^3 + x^2 + 3$ creates the string -50x^3 + 1x^2 + 3

$4x + 2x^5 - 3x^2 - 10$ reates the string 2x^5 - 3x^2 + 4x^1 – 10

<u>You may not use any built-in Java collections, as the underlying data structure</u>.