- **Zadanie 1**

  Stwórz klasę o nazwie *Tabele.java*, która będzie zawierała tabelę, co przechowuje liczby całkowite. Oprócz głównej metody ta klasa ma zawierać te oto statyczne metody:
  - Metodę *int sum (int[] arr)* która dodaje razem wszystkie elementy znajdujące się w tabeli i zwraca sumę.
  - Metodę *String toString (int[] arr)* która stworzy ciąg znaków, który reprezentuje zawartość tabeli. Ta metoda może być użyta w następujący sposób:
    ```
    int n = {3,4,5,6,7};
    String str = Arrays.toString(n);
    System.out.println("n = " + str);
    ```
  - Metodę *int[] addN (int[] arr, int n)* która stworzy nową tabelę, i doda do każdego elementu w tabeli liczbę *n*, i zwróci tą nową tabelę. Stara tabela *arr* ma pozostać taka sama.
  - Metodę *int[] reverse (int[] arr)* która stworzy nową tabelę, ta nowa tabela ma zawierać te same elementy co stara tabela, tylko że w odwróconej kolejności. Stara tabela ma pozostać taka sama.
  - Metodę *boolean hasN (int[] arr, int n)* która zwróci *true* jeśli *n* znajduje się w tabeli *arr*, inaczej zwróci *false*.
  - Metodę *void replaceAll (int[] arr, int old, int nw)* która zamieni wszystkie wartości *old*, na *nw*, wewnątrz tabeli *arr*.
  - Metodę *int[] sort (int[] arr)* która stworzy kopię tabeli *arr* i posortuje wartości wewnątrz tej nowej tabeli, od najmniejszej do największej. Stara tabela *arr* ma pozostać bez zmian.

  Twoja główna metoda powinna zawierać kod, który demonstruje jak działają napisane przez Ciebie metody.

  Musisz stworzyć te metody bez używania już wbudowanych metod z biblioteki Javy.

- **Zadanie 2**

  Stwórz klasę *Pesel.java* która oprócz głównej metody będzie zawiera kilka statycznych metod które będą dotyczyły numeru pesel. Załóżmy, że wszystkie numery pesel wyglądają wg. Tej formuły: *RRMMDD-XXXXX*. Metody statyczne powinny wykonywać następujące rzeczy:

  - Metodę *getFirstPart* która zawróci pierwszą część numeru pesel (*RRMMDD*).
  - Metodę *getSecondPart* która zawróci drugą część numeru pesel (*XXXXX*)

    Teraz załóżmy, że pierwsza litera w drugiej części numeru pesel (*XXXXX*) mówi nam czy dany pesel należy do kobiety lub mężczyzny. Jeśli ta pierwsza litera w *XXXXX* jest nieparzysta to jest to facet, a jeśli parzysta to jest to kobieta. Np. numer 810721-14831 należy do mężczyzny, bo na początku drugiej części numeru pesel (14831) jest cyfra nieparzysta. Tak więc stwórz następujące metody:

  - *isFemaleNumber* która zwróci *true* jeśli numer pesel należy do kobiety.
  - *isMaleNumber* która zwróci *true* jeśli numer pesel należy do mężczyzny
  - *areEqual* która porówna dwa numery pesel i zwróci *true* jeśli oba numery są identyczne.

- **Zadanie 3**
  Stwórz klasę *MultiDisplay.java* oraz klasę *MultiDisplayMain.java*. Klasa *MultiDisplayMain* ma zawierać główną metodę oraz taki kod:

  ```
  MultiDisplay md = new MultiDisplay();

  md.setDisplayMessage("Hello World!");
  md.setDisplayCount(3);
  md.display();                    // ==> print-out

  md.display("Goodbye cruel world!", 2);  // ==> print-out

  System.out.println("Current Message: "+ md.getDisplayMessage());
  ```

  Który przy uruchomieniu da taki oto rezultat:

  ```
  Hello World!
  Hello World!
  Hello World!
  Goodbye cruel world!
  Goodbye cruel world!
  Current Message: Goodbye cruel world!
  ```

  Klasa *MultiDisplay* musi oczywiście dać sobie radę z wyświetleniem różnych tekstów i różnych ilości wyświetleń.

- **Zadanie 4**
  W tym samym folderze co te ćwiczenia jest plik tekstowy o nazwie *AlarmClock*. Otwórz ten plik i weź kod który się tam znajduje, a następnie stwórz klasę *AlarmClock.java* który zawiera ten kod który znalazłem w tym pliku tekstowym.

  Stwórz teraz klasę *AlarmMain* który „używa” klasę *AlarmClock* do następujących rzeczy:
  1. Ustawia godzinę na 23:48
  2. Pokazuje czas
  3. Ustawia budzik na 6:15
  4. Pozwala aby czas upłynął, aby upłynęło 500 minut
  5. Pokazuje aktualny czas znowu

  Pamiętaj: Nie możesz nic zmienić w klasie AlarmClock, oprócz nazwy pakietu.

- **Zadanie 5**

  Stwórz klasę *TextAnalyzer* oraz *TextAnalyzerMain*. *TextAnalyzerMain* ma zawierać ten oto kod:

  ```
  String text = "My name is Anakin Skywalker. My age is 42.";
  TextAnalyzer ta = new TextAnalyzer(text);

  System.out.println("Char Count: "+ ta.charCount());
  System.out.println("Upper Case Count: "+ ta.upperCaseCount());
  System.out.println("Whitespace Count: "+ ta.whitespaceCount());
  System.out.println("Digit Count "+ ta.digitCount());
  ```

```
if (ta.containsChar('x'))
        System.out.println("The text contains char \'x\'");

if (ta.containsString("nakin"))
        System.out.println("The text contains substring \"nakin\"");
```

Który po uruchomieniu pokazuje następujący wydruk:

```
Char Count: 42
Upper Case Count: 4
Whitespace Count: 8
Digit Count 2
The text contains substring "nakin"
```

Klasa *TextAnalyzer* musi działać poprawnie także z innymi ciągami znaków.

- **Zadanie 6**

Napisz klasę *Point.java* oraz *PointMain.java*. Klasa *PointMain.java* ma zawierać kod:

```
Point p1 = new Point();
Point p2 = new Point(3,4);

System.out.println(p1.toString());  // ==> (0,0)
System.out.println(p2.toString());  // ==> (3,4)

if (p1.isEqualTo(p2))          // False!
   System.out.println("The two points are equal");

double dist = p1.distanceTo(p2);
System.out.println("Point Distance: "+dist);

p2.move(5,-2);       // ==> (8,2)
p1.moveToXY(8,2);    // ==> (8,2)

if (p1.isEqualTo(p2))          // True!
        System.out.println("The two points are equal");
```

Który po uruchomieniu program wyświetla:

```
(0,0)
(3,4)
Point Distance: 5.0
The two points are equal
```

Klasa *Point* musi też dawać sobie radę z innymi punktami (x,y). Pamiętaj:

- Koordynaty (x,y) są liczbami całkowitymi.
- Metoda *toString* zwraca ciąg znaków, który reprezentuje aktualne koordynaty punktu.

- Dystans pomiędzy dwiema punktami oblicza się w taki sam sposób jak w module 1, zadanie 14.
- Dwa punkty są identyczne (equal), jeśli mają takie same koordynaty.
- Metoda *move* przemieszcza punkt na osi x oraz osi y.
- Metoda *moveToXY* nadaje punktowi zupełnie nowe koordynaty.

- **Zadanie 7**

  Stwórz klasę *Fraction.java* która reprezentuje jakiś ułamek w formie L/M gdzie L (licznik) i M (mianownik) są liczbami całkowitymi. Jeśli mianownik byłby zerem to program ma pokazać odpowiedni komunikat. Klasa ta musi zawierać poniższe metody:

  - Konstruktor, który stwarza nowy ułamek.
  - Metodę *getNumerator* która zwraca licznik.
  - Metodę *getDenominator* która zwraca mianownik.
  - Metodę *isNegative* która zwraca true jeśli ułamek jest negatywny.
  - Metody *add*, *subtract*, *multiply*, *divide* które dodają, odejmują, mnożą oraz dzielą dwa ułamki ze sobą. Upewnij się że program odpowiednio zareaguje jeśli gdzieś wśród tych ułamków mianownik będzie zero.
  - *isEqualTo* która sprawdza czy dwa obiekty klasy *Fraction* reprezentują ten sam ułamek.
  - *toString* który zwraca ciąg znaków który pokazuje ułamek w formie L/M.

  Stwórz także klasę *FractionMain.java* która pokazuje jak działają wszystkie metody.

- **Zadanie 8**

  Wyobraź sobie talię 52 kart do gry. Jest to talia bez jokerów.

  Stwórz klasę *Card.java*, która reprezentuje jakąkolwiek kartę w takiej talii. Pamiętaj, że każda karta ma kolor (4 różne – trefl, karo, kier, pik) i rangę (13 możliwości – król, dama, itp.).

  Stwórz też klasę *Deck.java*, która ma reprezentować talię 52 kart oraz ma zawierać **52 obiekty** klasy *Card*.

  Klasa *Deck* ma posiadać metody, które odpowiadają za:
  - Pomieszanie kart
  - Rozdawanie kart
  - Poinformowanie użytkownika ile jeszcze kart zostało do rozdania
  - Jakie karty zostały rozdane

  Pamiętaj: Mieszanie kart ma być możliwe tylko wtedy, gdy żadne karty nie zostały jeszcze rozdane.

  Stwórz także klasę *DeckMain.java* która tworzy taką talię kart (czyli obiekt klasy Deck) i używa wszystkie metody które są zawarte w klasie Deck.