Zadanie 3.1: Losowe daty

Naszym celem jest stworzenie interaktywnego programu, który wykorzystuje kilka zaawansowanych mechanizmów z biblioteki standardowej Java, m. in. do reprezentacji danych czasowych (dat) i losowania.

Po wykonaniu wszystkich poleceń, program powinien działać następująco:

Na początku zadaje użytkownikowi pytanie:

```
Wybierz jedna z dostepnych opcji:
1 - recznie podana data
2 - losowa data
```

- Jeśli użytkownik wpisze `1`, program powinien zapytać kolejno dzień, miesiąc i rok. Jeśli data jest poprawna, program ją wypisze w odpowiednim formacie, jeśli nie - zgłosi błąd.
- Jeśli użytkownik wpisze `2`, program wylosuje za niego poprawną datę i ją wypisze.
- Jeśli użytkownik poda jakikolwiek inny numer, program powinien wypisać komunikat: `Nie ma takiej opcji.`

Na koniec program wypisze także rożnicę w latach między wskazaną datą, a datą dzisiejszą.

Uwaga: nie wolno ręcznie podawać dzisiejszej daty! Program powinien działać również gdy uruchomimy go za kilka lat.

Aby zrealizować zadanie, wykonaj pomocnicze polecenia i zapoznaj się z sekcją Przydatne informacje.

Polecenia

1. Korzystając z mechanizmów poznanych na poprzednich zajęciach, napisz program, który pyta użytkownika o dzień, miesiąc i rok i zapisuje te informacje w zmiennych typu `int`.
2. Skorzystaj z klasy `LocalDate` do utworzenia obiektu daty z wczytanych składowych. Wypisz utworzony obiekt.
3. Przenieś kod wczytujący datę do nowej metody, np. `loadUserDate()`. Metoda powinna zwracać obiekt typu `LocalDate`.
4. Stwórz drugą metodę `loadRandomDate()`, która również deklaruje typ zwracany `LocalDate`. Na razie metoda może być pusta - w środku umieść jedynie instrukcję `return null;`.
5. Za pomocą instrukcji `switch` dodaj do funkcji `main` obsługę opcji działania programu. Poszczególne opcje powinny używać przygotowanych metod `loadUserDate` i `loadRandomDate()`.

6. Korzystając z mechanizmu `Random` stwórz metodę `generateRandomNumber(int from, int to)`, która wylosuję liczbę całkowitą z podanego zakresu `<from, to>` i zwróci ją. Zanim napiszesz funkcję, wypróbuj działanie klasy `Random` - napisz kod losujący dowolną liczbę i uruchom kilkukrotnie program.
7. Zastosuj metodę `generateRandomNumber()` do stworzenia losowego dnia, miesiąca i roku w `loadRandomDate()`. Możemy założyć, że miesiąc ma 28 dni i że interesują nas lata od 1900 do 2015.
   Uzupełnij kod tak by metoda `loadRandomDate()` zwracała obiekt daty. W tym momencie program powinien już poprawnie działać i losować daty na żądanie użytkownika.
8. (*) Korzystając z metod `LocalDate` dodaj funkcji `main` wypisywanie różnicy między dzisiejszą datą, a wczytaną/wylosowaną. Co zrobić by różnica była zawsze dodatnia? Podpowiedź: można wykorzystać klasę `Math`: https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/lang/Math.html

Przydatne informacje

Zarówno w przypadku omawianych na zajęciach narzędzi, jak i wszelkich innych mechanizmów dostępnych w bibliotece standardowej Javy, Oracle udostępnia szczegółową dokumentację pakietów, klas i metod. Dokumentacja w formie API specification znajduje się na stronie (przykład dla Javy 13):

https://docs.oracle.com/en/java/javase/13/docs/api/index.html

W praktyce do znalezienia intersujących nas fragmentów dokumentacji najlepiej używać wyszukiwarki internetowej, np. Google.

Reprezentacja daty i czasu

W Javie 8 dodano obszerny zestaw narzędzi do reprezentacji daty i czasu. Wszystkie te narzędzia znajdują się w pakiecie java.time.

Najistotniejszą klasą z całego zestawu jest dla nas oczywiście `LocalDate`:
https://docs.oracle.com/en/java/javase/13/docs/api/java.base/java/time/LocalDate.html

Klasa ta nie posiada publicznego konstruktora, co oznacza, że nie możemy utworzyć jej obiektu za pomocą operatora `new`. Zamiast tego udostępnia ona wygodne, **statyczne** metody.
Metody statyczne to takie, które wywołujemy nie na konkretnym obiekcie klasy, a na samej klasie (globalnie). Powiemy o nich więcej na kolejnych zajęciach. Na razie potrzebujemy jedynie wiedzieć, jak stworzyć obiekt klasy `LocalDate`:

```
LocalDate date = LocalDate.of(2016, 11, 19);
```

Widzimy tu wywołanie statycznej metody, nazwanej po prostu `of`, która przyjmuje 3 parametry (rok, miesiąc, dzień).

Na tak utworzonym obiekcie daty możemy wykonywać różne operacje, np. sprawdzać czy rok był przestępny (metoda `isLeapYear()`) albo tworzyć nowe daty w oparciu o istniejące (dodając lata, miesiące itp.).

Mamy tu również specjalną metodę statyczną `now()`, która zwraca obiekt daty wskazujący na aktualny moment w czasie:

```
LocalDate currentDate = LocalDate.now();
```

Generator liczb pseudolosowych

W programowaniu często przydają nam się narzędzia do losowania danych. Java ma kilka takich narzędzi. Najpopularniejszym z nich jest klasa `Random`, znajdująca się w pakiecie `java.util`.

Aby otrzymać losową liczbę typu `int`, wystarczy utworzyć obiekt `Random` i wywołać na nim odpowiednią metodę:

```
Random random = new Random();
int randomNumber = random.nextInt();
int anotherRandomNumber = random.nextInt();
```

Każdorazowe wywołowanie `nextInt()` zwróci nam nową losową liczbę. Liczby losowanie są z całego dostępnego zakresu dla typu `int`. Jeśli chcemy ograniczyć losowanie do zakresu <0, N) powinniśmy użyć innej wersji tej metody:

```
int randomNumber = random.nextInt(N);
```

Uwaga: zakres jest domknięty z lewej, ale otwarty z prawej, tzn. że możemy wylosować 0, ale najwyższa możliwa liczba to N-1.

Instrukcja switch

Czasem zdarza się, że mamy do sprawdzenia kilka warunków polegających na dopasowaniu zmiennej do konkretnej wartości. Jeśli chcielibyśmy w takim przypadku użyć zwykłej instrukcji warunkowej `if`, musielibyśmy napisać:

```
int a = obliczA();
```

```
if (a == 1) {
  ...
} else if (a == 2) {
  ...
} else if (a == 3) {
  ...
}
```

Zamiast tego możemy zastosować instrukcję `switch`:

```
int a = obliczA();

switch(a) {
case 1:
    ...
    break;
case 2:
    ...
    break;
case 3:
    ...
    break;
}
```

`switch` próbuje dopasować wartość podanego wyrażenia do kolejnych wartości. Jeśli wartość pasuje, wykonywany jest kod znajdujący się pod instrukcją `case`. Instrukcja `break` przerywa sprawdzanie - teoretycznie nie jest obowiązkowa, ale gdybyśmy ją usunęli, `switch` po wykonaniu danego `case` nadal sprawdzałby kolejne przypadki (aż do ostatniego lub do napotkania `break`).

Oprócz konkretnych wartości, możemy także zdefiniować przypadek `default`, który oznacza "w pozostałych przypadkach". Jest to odpowiednik instrukcji `else` bez dodatkowego warunku `if`:

```
int a = obliczA();

switch(a) {
case 1:
    ...
    break;
case 2:
    ...
```

```
    break;
default:
    ...
    break;
}
```

Instrukcja `switch` działa dla danych typu `int`, ale również `char`, `byte`, `short` oraz `String` (obsługę Stringów dodano w Javie 7). W Javie 13 pojawiła się po raz pierwszy wersja przedpremierowa nowej funkcjonalności - tzw. `switch` expressions. Upraszczają one składnię i zwiększają jej aplikowalność. Więcej o tym można poczytać np. tutaj.