Przykładowe zadania programistyczne (MAS)

wersja z 2020-03-15

Wykładowca: dr inż. Mariusz Trzaska, (mtrzaska@pjwstk.edu.pl, http://www.mtrzaska.com)

Niniejsze zadania programistyczne mają na celu jedynie przypomnienie materiału dotyczącego programowania (przyswojonego w czasie wcześniejszych kursów) i ich rozwiązania nie będą oceniane w ramach przedmiotu MAS (Modelowanie i Analiza Systemów informacyjnych). Mogą być wykorzystane w czasie zajęć „*Wybrane konstrukcje obiektowych języków programowania*". Studenci przystępujący do kursu MAS, powinni umieć rozwiązać zdecydowaną większość poniższych zadań.

Dla każdego z zadań, przygotuj aplikację konsolową, np. w języku Java (GUI nie jest potrzebne) realizującą/ilustrującą poniższe wymagania. Każde rozwiązanie musi zawierać przykłady użycia, a co za tym idzie, odpowiednią ilość danych testowych. Zadbaj o właściwą jakość/czytelność/modyfikowalność kodu, m. in. nazewnictwo elementów, podział na mniejsze jednostki (metody, klasy, itp.). Upewnij się również, że kod jest poprawnie sformatowany.

1. Jak przechowasz wiele liczb typu `int` gdy wiadomo ile ich jest? Jak to zrobisz gdy w trakcie działania programu będą dodawane i/lub usuwane? Jak je wyświetlisz na konsoli?
2. Stwórz generator losowych string'ów dla podanej długości tekstu.
3. Stwórz generator tekstów typu *Lorem Ipsum*.
4. Napisz program zapamiętujący zmienną liczbę string'ów, a następnie usuwający losowe z nich. Wyświetl na konsolę dane przed i po operacji.
5. Wygeneruj zestaw losowych liczb z podanego zakresu. Następnie znajdź największą oraz najmniejszą spośród nich.
6. Zapamiętaj dane charakteryzujące graniastosłup prawidłowy. Opracuj rozwiązanie przechowujące dane o znanej oraz zmiennej liczbie graniastosłupów. Wyświetl na konsoli informacje ich dotyczące.
7. Zapamiętaj dane charakteryzujące dowolny graniastosłup. Opracuj rozwiązanie przechowujące dane o znanej oraz zmiennej liczbie graniastosłupów. Wyświetl na konsoli informacje ich dotyczące.
8. Przygotuj rozwiązanie modyfikujące wartość podanych liczb (np. zwiększające o 1).
9. Stwórz program obliczający liczbę dni od rozpoczęcia semestru oraz do jego zakończenia. Skorzystaj z klas (np. `LocalDateTime`) znajdujących się w pakiecie pakiet `java.time` dodanych do Java 8.
10. Przećwicz kilka refaktoringów/refaktoryzacji (*refactoring*) dostępnych w nowoczesnych IDE, np. w IntelliJ IDEA, np. *Safe Delete, Extract Method, Extract Constant, Extract Field, Extract Parameter, Introduce Variable, Rename*.
11. Podaj przykłady zastosowania konstrukcji `break` oraz `continue`.

12. Stwórz krótki program korzystający z konstrukcji `switch`. Pamiętaj o właściwym wykorzystaniu `break` oraz `default`.
13. Stwórz krótki program korzystający z pętli `while` oraz `do/while`. Czym one się różnią?
14. Chcemy przechować informacje o różnych rodzajach urządzeń, każde z nich ma różny zestaw cech. Jak to zrobić? Jak umieścić je we wspólnej kolekcji i wyświetlić na konsolę?
15. Mamy kompletnie różne byty, ale każdy z nich ma wspólną umiejętność, np. poruszania się. Jak to zapamiętać w programie, iterować po nich i uruchamiać tę umiejętność? Nie wolno stosować wspólnej nadklasy.
16. Podaj przykład wykorzystania własnej klasy obsługi błędu i wykorzystania jej razem z konstrukcją `throw`.
17. Załóżmy, że potrzebujemy przechować informacje, np. o silnikach. Mają jakieś wspólne cechy, a jedną z nich jest *rodzaj silnika*. Jak to zrobić? Jak będzie wyglądała metoda, która w zależności od *rodzaju silnika*, będzie robiła różne rzeczy. Podaj dwa sposoby rozwiązania tego problemu i ich zalety/wady.
18. Podaj przykłady wymagań biznesowych, które można zrealizować za pomocą różnych rodzajów pojemników. Stwórz takie implementacje.
    a. natywna tablica,
    b. lista,
    c. zbiór (*Set*),
    d. mapa (*Map*).

    W każdym z przypadków, dodaj kilka obiektów, usuń kilka spełniających określone wymagania, wyszukaj oraz wyświetl zawartość pojemników.
19. Po co w języku Java stosuje się *generics*? Stwórz przykładowe programy, wykorzystujące tę koncepcję przy implementacji klas oraz metod.
20. Podaj przykład zastosowania pętli *for-each*.
21. Zapisz do pliku losowe *string*'i. Następnie je odczytaj i wyświetl na konsoli.
22. Zapisz/odczytaj do/z pliku przykładowe dane opisujące osobę. Postaraj się aby zajmowały jak najmniej miejsca. Rozważ wykorzystanie klasy `ZipOutputStream` z biblioteki Java.
23. Wyświetl listę plików w podanym katalogu. Sprawdź czy znajduje się tam plik, którego nazwa zawiera podany tekst.
24. Podaj przykład wykorzystania konstrukcji *try-with-resources* języka Java.
25. Podaj przykład wykorzystania notacji diamentowej <> języka Java. Jakie są jej zalety/wady?
26. Stwórz przykład wykorzystujący umieszczenie kodu metody w interfejsie (wymaga Java 8 lub nowsza).
27. Stwórz przykład używający wyrażeń lambda w Java.
28. Stwórz kolekcję obiektów biznesowych (np. *Produkt*), a następnie:

a. wybierz z nich te, które spełniają przykładowe wymagania (np. cena wyższa od podanej kwoty),

b. policz ich sumaryczną wartość,

c. określ czy jest mniejsza od podanego progu.

Postaraj się aby kod był jak najbardziej zwięzły, ale zarazem czytelny. Rozważ skorzystanie z interfejsów funkcyjnych w pakiecie `java.util.function` (wymaga Java 8+).

29. Podaj przykłady zastosowania słowa kluczowego `var` w języku Java (wymaga Java 10+). Jak ma się ono do mocnej kontroli typologicznej oraz analogicznego słowa w *JavaScript*?

30. Przygotuj program porównujący wydajność różnego rodzaju pojemników (np. lista, mapa, zbiór) w operacjach wstawiania, dodawania, wyszukiwania oraz usuwania elementów.