Wyjątki

Ćwiczenie 1

Zadeklaruj 10 elementową tablicę liczb całkowitych. Wczytaj od użytkownika z klawiatury liczby do tablicy. Sprawdź co się stanie, gdy będziemy chcieli odczytać argument o indeksie 10. Obsłuż ten wyjątek korzystając z bloku `try {} catch() {}`.

Ćwiczenie 2

Utwórz klasę `Square` zawierającą metodę:

```
public static double square(int n);
```

obliczającą pierwiastek z liczby n. Wykorzystaj do obliczeń metodę `Math.sqrt();` Jeżeli jednak użytkownik przekaże liczbę mniejszą od 0 zwróć wyjątek `IllegalArgumentException`.

Ćwiczenie 3

Napisz klasę `Division`, która będzie posiadała metody:

```
public static double divide(int a, int b);
public static double divide(double a, double b);
```

Metoda `divide()` powinna zwracać `IllegalArgumentException` w przypadku podania 0 jako dzielną (zmienna b). Poinformuj kompilator o zwracanym błędzie.

Ćwiczenie 4

Utwórz klasę `ReadNumbers`, zawierającą metody:

```
public double readDouble();
public int readInt();
public String readString();
```

która wczyta od użytkownika odpowiedni format i zwróci wczytaną wartość wykorzystując klasę `Scanner` (z pakietu `java.util`). W przypadku wystąpienia wyjątku `InputMismatchException`, metody powinny zwracać kolejno 0.0, 0, "".

Ćwiczenie 5

Napisz klasę `QuadraticEquation` posiadającą konstruktor bez i sparametryzowany dla trzech pól:

```
private int a;
private int b;
private int c;
```

oraz metody:

`private int getNumber();` - wczytującą liczbę całkowitą od użytkownika

`public double[2] solve();` - rozwiązującą równanie kwadratowe

Metoda `solve()` w przypadku braku zainicjalizowania zmiennych `a, b, c` (lub podania `0` dla wszystkich trzech parametrów) powinna wywołać metodę `getNumber()` przed dokonaniem obliczeń. Metoda powinna zwracać tablicę z rozwiązaniami `[x1, x2]` bądź `[x1, x1]` i zgłaszać wyjątek `ArithmeticException` w przypadku ujemnej delty. Zwróć uwagę na przypadek, kiedy `a = 0, b !=0, c != 0`.

Obsłuż wyjątek w metodzie `main()`.

Ćwiczenie 6

Utwórz klasę `WrongAgeException`, która będzie rozszerzeniem klasy `Exception` i będzie posiadała dwa konstruktory (sparametryzowany oraz nie):

```
public WrongAgeException();
public WrongAgeException(String message);
```

obie metody powinny wywoływać konstruktor klasy nadrzędnej. W przypadku braku przekazania wiadomości, konstruktor powinien przekazywać informację `"Wrong age exception."`.

Ćwiczenie 7

Utwórz klasę `Person`, zawierającą pola:

```
private String name;
private String secondName;
private int age;
private String hair;
private String eyes;
private double shoe;
```

oraz `konstruktory` i `gettery` i `settery`. Zaimportuj klasę wyjątku z poprzedniego zadania. W przypadku podania wieku mniejszego równego `0` i większego równego `130` zwróć wyjątek. Obsłuż wyjątek w metodzie `main();`

Ćwiczenie 8

Wykorzystując klasę z poprzedniego zadania utwórz klasę `People`, która będzie przetrzymywała listę N osób (obiektów typu `Person`). Klasa powinna zawierać konstruktor sparametryzowany, którego parametr będzie wyznaczał ilość osób na liście oraz pola i metody:

```
private final String[] allowedEyes = { "green", "brown", "blue",
"black" };

private final String[] allowedHair = { "blond", "brown", "black",
"red" };
```

```
public void addPerson(Person person);

public void addPerson(String name, String secondName, int age,
String hair, String eyes, double shoe);
```

Druga metoda powinna sprawdzać czy przekazane argumenty są poprawne tj.

przekazane parametry `hair` oraz `eyes` zawierają się w tablicach predefiniowanych. Rozmiar buta może być wartością połówkową lub całkowitą (np. może przyjąć wartości: `9, 9.5, 10.50`, ale nie: `9.43`). W przypadku podania nieprawidłowych argumentów utwórz oraz zwróć wyjątki:

`BadHairException`, `BadEyesException`, `BadShoeException`.

Metody `addPerson()` dodatkowo powinny zwracać wyjątek `FullListException` (który należy utworzyć) w przypadku braku miejsca na liście.

Ćwiczenie 9

Napisz klasę `Puzzle`, która będzie zawierała predefiniowaną tablicę słów w polu oraz metody pozwalające na zgadywanie litery oraz zakończenie gry. Ilość możliwych prób zgadnięcia to `Math.ceil(1.5 * liczba_liter_w_slowie);` Pomyśl, w których momentach można dodać obsługę wyjątków (np. gry brak możliwości zgadywania, przekazany zły argument etc.) i ewentualnie utwórz wyjątki i dodaj je do klasy `Puzzle`. Metoda zgadująca powinna zgłaszać wyjątek gdy brak możliwości odgadywania bądź prosić użytkownika o podanie litery. Przy podaniu litery metoda powinna drukować zamaskowane słowo np.

```
Ilość prób: 8/12

O K S Y _ O _ O _


Podaj literę:
```