Politechnika Krakowska
przedmiot: Systemu Odporne na Błędy
Laboratorium 2: Asercja w języku Java
sprawozdanie

wykonał: Tomasz Fudalej

1. **Asercja** (ang. assertion) – predykat (forma zdaniowa w danym języku, która zwraca prawdę lub fałsz), umieszczony w pewnym miejscu w kodzie. Asercja wskazuje, że programista zakłada, że predykat ów jest w danym miejscu prawdziwy. W przypadku gdy predykat jest fałszywy (czyli niespełnione są warunki postawione przez programistę) asercja powoduje przerwanie wykonania programu.

2. **Zadanie:** sprawdzenie działania asercji zostanie wykonane przy pomocy klasy *NumberSet* przechowującej zbiór liczb typu całkowitego, która posiada następujące metody:
   - DODANIE NOWEJ WARTOŚCI
   - USUNIĘCIE WYBRANEJ WARTOŚCI
   - USUNIĘCIE LOSOWEJ WARTOŚCI I ZWRÓCENIE JEJ
   - ZWRÓCENIE SUMY ELEMENTÓW ZBIORU
   - PODZIELENIE KAŻDEGO ELEMENTU PRZEZ DANĄ LICZBĘ
   - SPRAWDZENIE, CZY DANY ELEMENT ZNAJDUJE SIĘ W ZBIORZE
   - ZWRÓCENIE ROZMIARU ZBIORU

3. **Implementacja i testowanie funkcji**:
   1. DODANIE NOWEJ WARTOŚCI:
      1.1. Implementacja:

```
public void add(int i) throws Exception{
    System.out.println("Add [" + i + "]");

    assert(size >= 0 && size <= MAX_SIZE) : "Niedozwolony rozmiar zbioru danych. "; //1)

    if(i < MIN_VAL && i > MAX_VAL)
        throw new Exception("Dodawana wartość [" + i + "] nie mieści się w zakresie zbioru."); //2)

    if(size < MAX_SIZE){
        nSet[size++] = i;
    }
    else if(size == MAX_SIZE)
        throw new Exception("Zbiór pełny."); //3)
    else {
        assert(size > MAX_SIZE) : "Zbiór przepełniony."; //4)
    }
}
```

      1.2. Wyjaśnienie użytych metod wykrywania błędów:
      1) Asercja, ponieważ przekroczenie przez zmienną *size* podanego zakresu oznacza błąd krytyczny, uniemożliwiający poprawne działanie programu.
      2) Wyjątek, ponieważ próba dodania niepoprawnej danej nie uniemożliwia dalszego działania programu.
      3) Wyjątek, ponieważ zbiór jest pełny, ale jest to stan dozwolony.
      4) Asercja, ponieważ zbiór przekroczył dozwolony rozmiar.

      1.3. Testowanie:
      a) Przepełnienie zbioru powoduje wyrzucenie wyjątku:

```
MAX_SIZE = 5
Add [1]
Add [2]
Add [3]
Add [4]
Add [5]
Add [6]
java.lang.Exception: Zbiór pełny.
        at number.set.NumberSet.add(NumberSet.java:32)
        at number.test.NSTest.main(NSTest.java:21)
```

b) Wymuszone za pomocą kodu testującego przekroczenie zakresu rozmiaru zbioru (naśladujące np. błąd sprzętowy) powoduje reakcję asercji:

```
MAX_SIZE = 5
Add [1]
Add [2]
Add [3]
Add [4]
Add [5]
Add [6]
Exception in thread "main" java.lang.AssertionError: Niedozwolony rozmiar zbioru danych.
        at number.set.NumberSet.toString(NumberSet.java:179)
        at java.lang.String.valueOf(String.java:2994)
        at java.io.PrintStream.println(PrintStream.java:821)
        at number.test.NSTest.main(NSTest.java:22)
```

2. USUNIĘCIE WYBRANEJ WARTOŚCI:

2.1. Implementacja:

```
public void remove(int i) throws Exception {
    System.out.println("Remove [" + i + "]");

    assert(nSet != null) : "Nastąpiło niedozwolone usunięcie zbioru danych."; //1)

    int removed = 0;
    for(int j = 0; j < size; j++){
        if(nSet[j] == i){
            nSet[j] = 0;
            removed++;
        }
        else if(removed > 0){
            nSet[j-removed] = nSet[j];
        }
    }
    size -= removed;
    assert(size >= 0 &&  size <= MAX_SIZE); //2)

    if(removed == 0)
        throw new Exception("Brak liczby w zbiorze.");  //3)
}
```

2.2. Wyjaśnienie użytych metod wykrywania błędów:

1) Asercja, ponieważ tablica *nSet* jest inicjalizowana przy inicjalizowaniu klasy *NumberSet* i nie powinna być usunięta do końca życia instancji klasy. Przypisanie jej wartości *null* jest błędem krytycznym.
2) Asercja, ponieważ *size* przekroczył dozwoloną wartość.
3) Wyjątek, ponieważ brak elementu w zbiorze nie uniemożliwia działania programu

2.3. Testowanie:

a) Gdy zbiór został usunięty wywołana jest asercja:

```
nSet == null
Remove [2]
Exception in thread "main" java.lang.AssertionError: Nastąpiło niedozwolone usunięcie zbioru danych.
        at number.set.NumberSet.remove(NumberSet.java:53)
        at number.test.NSTest.main(NSTest.java:25)
```

b) Próba usunięcia niebędącego w zbiorze elementu powoduje wyrzucenie wyjątku:

```
Add [1]
Add [3]
Add [4]
Add [5]
Data Set = 1, 3, 4, 5,
Remove [2]
java.lang.Exception: Brak liczby w zbiorze.
        at number.set.NumberSet.remove(NumberSet.java:69)
        at number.test.NSTest.main(NSTest.java:23)
```

3. USUNIĘCIE LOSOWEJ WARTOŚCI I ZWRÓCENIE JEJ

3.1. Implementacja:

```
public int getRandomValue() throws Exception {
    assert(nSet != null) : "Nastąpiło niedozwolone usunięcie zbioru danych."; //1)
    assert(size >= 0 && size <= MAX_SIZE) : "Niedozwolony rozmiar zbioru danych. "; //2)

    if(size == 0)
        throw new Exception("Zbiór jest pusty."); //3)

    Random random = new Random();
    int genNum = random.nextInt(MAX_VAL) + MIN_VAL;

    for(int i = 0; i < size; i++){
        if(nSet[i] == genNum){
            int val = nSet[i];
            remove(nSet[i]);
            return val;
        }
    }

    throw new Exception("Brak liczby [" + genNum + "] w zbiorze."); //4)
}
```

3.2. Wyjaśnienie użytych metod wykrywania błędów:

1) i 2) Dwie pierwsze asercje sprawdzają opisywane przy wcześniejszych funkcjach błędy krytyczne.

3) Wyjątek z powodu braku danych w zbiorze.

4) Wyjątek z powodu braku poszukiwanej wartości w zbiorze.

3.3. Testowanie:

a) wyjątek dotyczący pustego zbioru:

```
Data Set =
java.lang.Exception: Zbiór jest pusty.
        at number.set.NumberSet.getRandomValue(NumberSet.java:84)
        at number.test.NSTest.main(NSTest.java:28)
```

b) wyjątek dotyczący braku szukanego elementu

```
Data Set = 1, 3, 4, 5,
java.lang.Exception: Brak liczby [7] w zbiorze.
        at number.set.NumberSet.getRandomValue(NumberSet.java:97)
        at number.test.NSTest.main(NSTest.java:28)
```

4. ZWRÓCENIE SUMY ELEMENTÓW ZBIORU

4.1. Implementacja:

```
public int getSumOfElements() throws Exception {
    assert(nSet != null) : "Nastąpiło niedozwolone usunięcie zbioru danych."; //1)
    assert(size >= 0 && size <= MAX_SIZE) : "Niedozwolony rozmiar zbioru danych. "; //2)
    if(size == 0)
        throw new Exception("Zbior jest pusty."); //3)

    int sum = 0;
    for(int i = 0; i < size; i++)
        sum += nSet[i];
    assert(sum >= MIN_VAL * size && sum <= MAX_VAL * size) : "Niedozwolona suma elementów."; //4)
    return sum;
}
```

4.2. Wyjaśnienie użytych metod wykrywania błędów:

1) i 2) Dwie pierwsze asercje sprawdzają opisywane przy wcześniejszych funkcjach błędy krytyczne.

3) Wyjątek z powodu braku danych w zbiorze.

4) Asercja sprawdzająca, czy suma elementów mieści się w przedziale poprawnych wartości.

4.3. Testowanie:

a) wymuszone w kodzie testującym przekroczenie zakresu poprawnych wyników sumy powoduje zasygnalizowanie błędu przez asercję:

```
Data Set = 1, 2, 3, 4, 5,
SUM = 1014
Exception in thread "main" java.lang.AssertionError: Niedozwolona suma elementów.
        at number.set.NumberSet.getSumOfElements(NumberSet.java:119)
        at number.test.NSTest.main(NSTest.java:33)
```

5. PODZIELENIE KAŻDEGO ELEMENTU PRZEZ DANĄ LICZBĘ

5.1. Implementacja:

```
public void divideAllElementsBy(int d) throws Exception {
    assert(d != 0) : "Wystąpiło dzielenie przez zero."; //1)
    assert(nSet != null) : "Nastąpiło niedozwolone usunięcie zbioru danych."; //2)
    assert(size >= 0 && size <= MAX_SIZE) : "Niedozwolony rozmiar zbioru danych. "; //3)

    try{
        for(int i = 0; i < size; i++){
            nSet[i] = nSet[i] / d;
            assert(nSet[i] >= MIN_VAL && nSet[i] <= MAX_VAL) :
                "Dzielenie spowodowało przekroczenie zakresu wartości w zbiorze."; //4)

        }
    }catch(ArithmeticException e){
        throw new Exception("Dzielenie przez zero."); //5)
    }
}
```

5.2. Wyjaśnienie użytych metod wykrywania błędów:

1) i 5) sprawdzają ten sam rodzaj błędu. Obie metody są tutaj dopuszczalne, ponieważ w przypadku wykrycia niedozwolonej operacji, nie zostanie ona wykonana i zbiór danych

pozostanie nieuszkodzony. Można więc wyobrazić sobie sensowne obsłużenie takiej sytuacji przez zwykła obsługę wyjątku, np. pominięcie całej funkcji.

5.3. Testowanie:

a) poprawne działanie:

```
Dividing by: 2
Before: Data Set = 1, 2, 3, 4, 5,
After: Data Set = 0, 1, 1, 2, 2,
```

b) wykrycie dzielenia przez zero przy pomocy asercji:

```
Exception in thread "main" java.lang.AssertionError: Wystąpiło dzielenie przez zero.
        at number.set.NumberSet.divideAllElementsBy(NumberSet.java:130)
        at number.test.NSTest.main(NSTest.java:36)
```

c) wykrycie dzielenia przez zero przy pomocy wyjątku:

```
java.lang.Exception: Dzielenie przez zero.
        at number.set.NumberSet.divideAllElementsBy(NumberSet.java:142)
        at number.test.NSTest.main(NSTest.java:36)
```

6. SPRAWDZENIE, CZY DANY ELEMENT ZNAJDUJE SIĘ W ZBIORZE

6.1. Implementacja:

```
public boolean contains(int i) throws Exception {
    if(i < MIN_VAL || i > MAX_VAL)
        throw new Exception ("Szukana wartość [" + i + "] nie mieści się w zakresie zbioru."); //1)
    assert(nSet != null) : "Nastąpiło niedozwolone usunięcie zbioru danych."; //2)
    assert(size >= 0 && size <= MAX_SIZE) : "Niedozwolony rozmiar zbioru danych. "; //3)

    for(int j = 0; j < size; j++){
        if(nSet[j] == i)
            return true;
    }
    return false;
}
```

6.2. Wyjaśnienie użytych metod wykrywania błędów:

1) Nieudana próba wyszukania wartości w zbiorze nie powoduje zatrzymania działania programu.

2) i 3) Dwie ostatnie asercje sprawdzają opisywane przy wcześniejszych funkcjach błędy krytyczne.

6.3. Testowanie:

a) poprawne działanie:

```
Data Set = 1, 2, 3, 4, 5,
Element 3 znajduje się w zbiorze?
true
```

b) Wyjątek sygnalizujący poszukiwanie wartości spoza zakresu:

```
java.lang.Exception: Szukana wartość [20] nie mieści się w zakresie zbioru: <0, 10>.
        at number.set.NumberSet.contains(NumberSet.java:160)
        at number.test.NSTest.main(NSTest.java:41)
```

7. ZWRÓCENIE ROZMIARU ZBIORU

7.1. Implementacja:

```
public int getSize() {
    assert(size >= 0 && size <= MAX_SIZE) : "Niedozwolony rozmiar zbioru danych. "; //1)
    return size;
}
```

7.2. Wyjaśnienie użytych metod wykrywania błędów:

Błąd krytyczny przekroczenia dozwolonego rozmiaru zbioru.

7.3. Testowanie:

```
Exception in thread "main" java.lang.AssertionError: Niedozwolony rozmiar [20] zbioru danych. MAX_SIZE = 5
        at number.set.NumberSet.getSize(NumberSet.java:177)
        at number.test.NSTest.main(NSTest.java:45)
```