3 Testowanie oparte na specyfikacji (czarnoskrzynkowe)

Pojęcia: czarnoskrzynkowe techniki projektowania testów, podział na klasy równoważności, charakterystyka, analiza wartości brzegowych, testowalność, tablice decyzyjne, Category-Partition, CRUD, analiza dziedzinowa, punkty ON, OFF, IN, OUT, testowanie kombinacyjne, tablice ortogonalne, maszyna stanowa, testowanie oparte o słowa kluczowe, testowanie eksploracyjne.

3.1 Wyznaczanie klas równoważności i wartości brzegowych

Dla każdego z poniższych zadań wyznacz – jeśli to możliwe – klasy równoważności dla analizowanej zmiennej, zarówno poprawne jak i niepoprawne, a następnie wybierz wartości do testów i wyznacz oczekiwaną odpowiedź systemu dla każdego z testów stosując techniki: 1) podziału na klasy równoważności, 2) dwóch wartości granicznych, 3) trzech wartości granicznych.

A) System obsługi pasażerów przyznaje zniżki na bilety w zależności od wieku pasażera. Dzieciom do lat 6 przysługują przejazdy bezpłatne, a emerytom (osobom w wieku powyżej 65 lat) przysługuje zniżka 50%.

B) System nalicza podatek w zależności od wysokości dochodu, podanego w pełnych złotych, w następujący sposób: dochody do 8,000 PLN włącznie są nieopodatkowane. Dla dochodów wyższych, ale nie przekraczających 70,000 PLN włącznie podatek wynosi 10%. Dla dochodów powyżej 70,000 ale niższych od 200,000 PLN podatek wynosi 7,000 plus 20% od kwoty powyżej 70,000 PLN. Dochody powyżej 200,000 PLN są opodatkowane kwotą 33,000 PLN plus 30% od kwoty powyżej 200,000 PLN.

C) Formularz zamówienia w sklepie internetowym zawiera pole "Nazwisko", które – w celu dokonania zakupu – musi być poprawnie wypełnione, tzn. w szczególności musi być niepuste.

D) Formularz zawiera pole "PESEL", system udziela zniżki osobom poniżej 18 i powyżej 65 lat.

3.2 Podział na klasy równoważności i analiza wartości brzegowych

Ściągnij skompilowaną wersję programu *Triangle*. Program ten przyjmuje na wejściu trzy liczby całkowite, a następnie zwraca informacje, czy z trzech odcinków o podanych długościach da się zbudować trójkąt, a ponadto czy da się z nich zbudować trójkąt równoboczny, równoramienny oraz różnoboczny.

Na podstawie powyższej specyfikacji zaprojektuj przy pomocy technik czarnoskrzynkowych: podziału na klasy równoważności oraz analizy wartości brzegowych testy dla programu *Triangle*, wykonaj je ręcznie i sprawdź wyniki. Zanim przystąpisz do projektowania i wykonywania testów:

A) Dokonaj analizy dziedziny (zarówno wejściowej jak i wyjściowej) – jakie występują zmienne (jawne i niejawne), jakie wartości mogą przyjmować, jakie są poprawne a jakie niepoprawne wartości tych zmiennych? Określ charakterystyki, pod których kątem będziesz rozważał testowanie programu *Triangle*. Charakterystyki mogą być oparte zarówno na składni (interfejsie), jak i na funkcjonalności (semantyce działania programu).

B) Zidentyfikuj klasy równoważności dla każdej charakterystyki i dokonaj ich podziału, a następnie wybierz reprezentantów do testów.

C) Zidentyfikuj wartości brzegowe dla tych klas równoważności, dla których jest to możliwe. Wyznacz elementy pokrycia dla metod: dwóch oraz trzech wartości granicznych.

D) Po wykonaniu testów zastanów się, czy przyjęta przez Ciebie strategia podejścia do testów byłaby inna, gdyby interfejs programu miał inną postać? Np. gdyby wartości wejściowe były przekazywane przez formularz na stronie www? Albo były wybierane z rozwijalnej listy? Albo wpisywane z listy poleceń jako parametry wywołania programu? Jak postać interfejsu mogłaby wpłynąć na testowanie?

3.3 Klasy równoważności dla większej liczby charakterystyk

Poniżej znajduje się uproszczona specyfikacja rzeczywistego polecenia DIMENSION, służącego do definiowania w Fortranie tablicy. W opisach składni kursywą zaznaczono miejsca, pod które należy obowiązkowo podstawić określony element, nawiasy kwadratowe obejmują części opcjonalne, a wielokropek oznacza, że część bezpośrednio przed nim następująca może występować więcej niż raz.

[POCZĄTEK SPECYFIKACJI]
Polecenie DIMENSION jest używane do specyfikowania wymiarów tablicy. Postać polecenia jest następująca:

DIMENSION *ad*[,ad]...

gdzie *ad* jest tzw. deskryptorem tablicy (array descriptor) postaci

n(d[,d]...)

gdzie *n* jest symboliczną nazwą tablicy a *d* jest deklaratorem wymiaru. Nazwy symboliczne mogą mieć od jednego do sześciu znaków – liter lub cyfr, przy czym nazwa nie może rozpoczynać się cyfrą. Minimalna i maksymalna liczba deklaratorów wymiaru dla definiowanej tablicy to odpowiednio jeden i siedem. Deklarator wymiaru ma postać:

[lb:]ub

gdzie *lb* i *ub* to odpowiednio dolna i górna granica wymiaru (lower bound, upper bound). Granica może być stała, w zakresie od –65534 do 65535 lub też może być nazwą zmiennej typu integer (ale nie nazwą elementu tablicy). Jeśli *lb* nie jest wyspecyfikowane, zakłada się, że wynosi 1. Wartość *ub* musi być większa lub równa *lb*. Jeśli *lb* jest wyspecyfikowane, jego wartość może być ujemna, dodatnia lub równa zeru. Polecenie DIMENSION może rozciągać się na więcej niż jedną linię.
[KONIEC SPECYFIKACJI]

A) Zidentyfikuj charakterystyki.
B) Wyprowadź klasy równoważności ze zidentyfikowanych charakterystyk. Wyróżnij klasy poprawne i klasy niepoprawne. Każdej klasie przypisz unikatowy identyfikator (1, 2, 3 itd.)
C) Stwórz przypadki testowe zgodnie z przyjętą zasadą, że najpierw staramy się pokryć pojedynczym przypadkiem testowym jak najwięcej klas poprawnych, a potem dla każdej klasy niepoprawnej tworzymy osobny przypadek testowy.

3.4 Testowalność aplikacji

Testowanie programu *Triangle* w sposób taki jak w zadaniu 3.2 jest dosyć uciążliwe i nieefektywne. Zautomatyzuj proces testowania w następujący sposób:

A) Zrefaktoryzuj kod programu *Triangle.java* tak, aby długości odcinków przyjmował jako parametry wejścia (przy uruchamianiu programu).

B) Zrefaktoryzuj kod programu *Triangle.java* tak, aby wynik zwracany był w postaci jednej liczby (np.: 0=nie trójkąt, 1=równoboczny, 2=równoramienny i nie równoboczny, 3=różnoboczny).

C) Zaprojektowane przez Ciebie przypadki testowe w zadaniu 3.2 przepisz do pliku `testy.txt`. Każdy nieparzysty wiersz tego pliku zawierać ma parametry wejścia do programu *Triangle.java*, a każdy wiersz parzysty – oczekiwany wynik zwrócony przez program dla parametrów z linii powyżej, na przykład:

```
6 6 6
1
4 5 4
2
4 11 999
0
```

D) Napisz skrypt (w dowolnym języku), który będzie uruchamiał program *Triangle* dla danych wejściowych z pliku `testy.txt` oraz porównywał w sposób automatyczny zwrócony przez program wynik z wynikiem oczekiwanym; skrypt jako wynik swojego działania powinien zwracać następujące informacje: zestaw parametrów, wynik oczekiwany, wynik rzeczywisty (dla każdego przypadku testowego z pliku `testy.txt`), oraz – na sam koniec – podsumowanie: liczba testów uruchomionych, liczba testów zdanych oraz liczba testów niezdanych.

E) Refaktoryzacja programu *Triangle.java* spowodowała, że wejście do programu podawane jest w inny sposób – z linii poleceń; czy wpływa to w jakiś sposób na zmianę podejścia do testowania? Czy można dodać do zestawu stworzonych przez Ciebie w Zadaniu 1 testów dodatkowe testy, sprawdzające więcej możliwości? Przedyskutuj ten problem w grupie (tzn. jak sposób podawania wejścia może wpływać na podejście do testowania).

F) jeśli możesz, dodaj więcej nowych testów do pliku `testy.txt` i uruchom swój skrypt testowy.

UWAGA – zachowaj plik `testy.txt` ze stworzonymi przez Ciebie testami – testy te wykorzystasz na jednych z późniejszych ćwiczeń.

3.5 Analiza wartości brzegowych

Rozważ moduł *CarPanel*, który na podstawie dwóch danych wejściowych s oraz t typu *double* określających przebytą drogę (w metrach, z dokładnością do centymetra) oraz czas (w sekundach, z dokładnością do setnej sekundy) wyświetla na panelu kierowcy aktualną prędkość v (w km/h, zmienna typu *double*). Jeśli czas pomiaru wynosił 0, podawana prędkość wynosi 0. Jeśli prędkość jest większa niż 120 km/h, panel wyświetla ostrzeżenie o przekroczeniu prędkości (poprzez zmianę ustawienia flagi $WARN$ z 0 na 1). Dane dotyczące drogi i czasu pobierane są z zewnętrznego czujnika nie rzadziej niż raz na sekundę. Chcemy przetestować ten program przy pomocy analizy wartości brzegowych stosując metodę dwóch wartości granicznych.

A) Wyprowadź warunki testowe.
B) Wyprowadź elementy pokrycia stosując technikę dwóch wartości granicznych.
C) Zdefiniuj przypadki testowe.