1. Co to są i po co są asercje w programowaniu?

Asercja to 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. Asercja ma szczególne zastosowanie w trakcie testowania tworzonego oprogramowania, np. dla sprawdzenia luk lub jego odporności na błędy. Zaletą stosowania asercji jest możliwość sprawdzenia, w którym fragmencie kodu źródłowego programu nastąpił błąd.

2. Co to są i po co są adnotacje w programowaniu?

Adnotacja to składniowa forma metadanej, która może zostać dodana do kodu źródłowego. Mogą być dodawane do definicji klasy albo deklaracji jej składowych w następującej formie: @adnotacja. Nie mają bezpośredniego wpływu na działanie programu, ale wpływają na sposób jego obsługi przez kompilator oraz rozmaite narzędzia dodane do kompilatora.

Rodzaje adnotacji :

* @Deprecated – oznacza, że danego elementu (metoda, klasa, zmienna) nie należy używać, ponieważ nie jest zalecany (np. przestarzałe metody)
* @Override – może być używana tylko przed deklaracją metody i informuje kompilator, że dana metoda przesłania metodę klasy bazowej
* @SuppressWarnings – wstrzymywanie ostrzeżeń

3. Co to są i po co są dzienniki w programowaniu?

Dziennik to w chronologiczny zapis zawierający informację o zdarzeniach i działaniach dotyczących systemu komputerowego Np. faktu uruchomienia lub logowania się do komputera, uzyskania połączenia sieciowego, awarii drukarki itp. Służy np. do rejestracji błędów podczas wykonywania pracy programów, do zapisów zmian w bazach danych

4. Co to są testy czarno-skrzynkowe w programowaniu?

* Tester otrzymuje aplikację lub jej moduł i stara się ją doprowadzić do niepoprawnego działania
* Nie wie jak testowany element jest zbudowany, a tylko wie jaką logikę powinien spełniać.
* Zaleta: Możliwość wyszukania nietypowych zachować użytkownika. – Testy integracyjne, testy interfejsu użytkownika, testy akceptacyjne.
* Wada: niska automatyzacja procesu testowego

5. Co to są testy biało-skrzynkowe w programowaniu?

* Zazwyczaj są przeprowadzane wewnątrz zespołu programistów.
* Mogą one obejmować zarówno całe moduły i być traktowane jako testy obciążeniowe/wydajnościowe, ale mogą też dotyczyć niewielkich fragmentów kodu takich jak pojedyncze klasy czy metody (testy jednostkowe)

6. Co to jest "słabe debuggowanie"?

Jest to wypisywanie na standardowe wyjście stanu systemu w pewnych miejscach (System.out.println()).

Szeroko stosowane przez osoby początkujące jak i profesjonalistów

Wady: konieczność kompilacji kodu za każdym razem gdy wykonujemy test oraz po zakończeniu testu i usunięciu metod wypisujących (kompilacja kodu po zakończeniu testów oznacza, że może się on zmienić! )

7. Co to jest i jak działa debugger?

Program komputerowy służący do dynamicznej analizy innych programów, w celu odnalezienia i identyfikacji zawartych w nich błędów, zwanych z angielskiego bugami (robakami). Proces nadzorowania wykonania programu za pomocą debuggera określa się mianem debugowania.

Podstawowym zadaniem debuggera jest sprawowanie kontroli nad wykonaniem kodu, co umożliwia zlokalizowanie instrukcji odpowiedzialnych za wadliwe działanie programu. Współczesne debuggery pozwalają na efektywne śledzenie wartości poszczególnych zmiennych, wykonywanie instrukcji krok po kroku czy wstrzymywanie działania programu w określonych miejscach. Debugger jest standardowym wyposażeniem większości współczesnych środowisk programistycznych.

Debuggery posiadają również wady – symulacja działania kodu nie jest idealnym odtworzeniem wykonania tego kodu w warunkach normalnych. Wobec tego debuggery mogą nie wykrywać bugów niezależnych bezpośrednio od treści badanego programu.

8. Omówić cechy dobrych testów jednostkowych.

* Szybkość: Testy jednostkowe powinny uruchamiać się szybko, tak aby nie opóźniały znacząco wykonanie builda. Szybko oznacza dużo poniżej 1 s.
* Izolacja: Testy powinny być od siebie odizolowane i niezależne od siebie. Test nie powinien uruchamiać innego testu.
* Powtarzalność: Testy powinny być powtarzalne na każdym środowisku. Testy nie mogą mieć stanów początkowych, ani zasobów do wyczyszczenia. Oznacza to także brak zależności w stosunku do zasobów zewnętrznych (baza danych, system plików, itd.)
* Zgodność: Testy powinny dawać ten sam rezultat za każdym uruchomieniem.
* Atomiczność: Atomiczne testy oznaczają, że test jednoznacznie jest zielony lub czerwony. Nie ma przypadku, gdy test jest zielony, ale testowana logika nie działa prawidłowo z jakichś powodów. Nie ma testów częściowo poprawnych.
* Asercja: Każdy test powinien mieć co najmniej jedną asercję. To jest największa oczywistość, ale należy też pamiętać o tym, że test bez asercji przechodzi jako zielony (poza przypadkiem kiedy testowany kod wyrzuci wyjątek). Ktoś może napisać test bez asercji, aby sprawdzić czy jego kod nie wyrzuci wyjątku. Framework do testowania powinien mieć oddzielną metodę w klasie Assert do sprawdzania czy wyjątek został lub nie został wyrzucony przez zadany kod.
* Rostrzygalność: Oprócz stanu czerwony i zielony, testy mogą mieć też stan nierozstrzygnięty (ang. inconclusive). Taki stan oznacza, że nie udało się rozstrzygnąć czy dana asercja jest spełniona lub nie.
* Zasada pojedynczej odpowiedzialności: Jeden test jednostkowy powinien testować jedną logiczną asercję (lub inaczej: jedno zachowanie). Oznacza to, że nie umieszczamy wielu asercji testujących różne zachowania w jednym teście jednostkowym. Musimy podzielić taki test na n metod (n - liczba asercji) lub wprowadzić testy sparametryzowane. Ta zasada nie oznacza, że jeden test powinien mieć tylko jeden Assert. Wiele wywołań Assert może przecież testować jedną logicznę asercję. Przykładem użycia wielu metod klasy Assert w obrębie jednej logicznej asercji może być asercja sprawdzająca poprawność elementów kolekcji.
* Niezależność: Wszystkie zależności wewnętrzne (np. zależne klasy, interfejsy) oraz zewnętrzne (np. baza danych, system, sieć wewnętrzna, Internet, web service) powinny być zastąpione przez test doubles (stuby, fake-i, mocki,itd.) Dzięki niezależności zyskujemy nie tylko oddzielenie się od implementacji konkretnej zależności, ale także szybkość i reużywalność testu.
* Second Class Citizens: Kod testowy nie jest kodem drugiej kategorii. Należy o niego dbać, aktualizować, refaktoryzować, robić review w taki sam sposób jak kod produkcyjny.

9. Co to jest i do czego służy imitacja w programowaniu?

Imitacje obiektów - zastępują obiekty rzeczywiste na czas uruchamiania i testowania kodu

* Problem: zbyt duży koszt implementacji imitacji obiektów

10. Co to jest i do czego służy atrapa w programowaniu?

Symulowany obiekt, który w kontrolowany sposób naśladuje zachowanie rzeczywistego obiektu. Programista tworzy zazwyczaj atrapy obiektów w celu przetestowania zachowania jakiegoś innego obiektu, podobnie jak projektanci samochodów wykorzystują manekiny do symulacji dynamiki zachowania ludzkiego ciała podczas zderzenia pojazdów.

W testach jednostkowych, atrapy obiektów mogą symulować zachowanie złożonych obiektów rzeczywistych, co jest przydatne, gdy rzeczywistego obiektu nie można użyć w teście jednostkowym (lub gdy jest to niepraktyczne). Jeśli obiekt ma jedną z poniższych cech, warto rozważyć użycie w jego miejsce atrapy:

* zwracanie niedeterministycznych wyników (np. aktualnego czasu lub temperatury),
* ma stany, które są trudne do wywołania lub zreplikowania (np. błąd sieciowy),
* jest powolny (np. kompletna baza danych, wymagająca inicjalizacji przed testem),
* jeszcze nie istnieje lub jego zachowanie może się zmienić,
* wymagałby dołączenia informacji i metod przeznaczonych wyłącznie do testów, a nie do realizacji celu, dla którego powstał.

Na przykład program budzika uruchamiający dzwonek o określonej godzinie może pobierać aktualny czas z zewnątrz. Aby go przetestować test musiałby oczekiwać na nastanie godziny alarmu, aby zweryfikować, czy budzik zadzwoni. W tej sytuacji użycie atrapy obiektu zaprogramowanej tak, aby zwracała godzinę alarmu jako aktualny czas niezależnie od tego, czy jest to właściwa godzina, czy nie, pozwoliłoby przetestować program budzika w izolacji.