Sun Certified Java Programmer

SCJP 6.0

Harmonogram szkolenia

wykonał:

mgr inż. Mateusz Maksymiuk

Sun Certified Java Programmer 5.0
Sun Certified Java Programmer 6.0
Sun Certified Java Developer
Sun Certified Web Component Developer 1.4
Sun Certified Business Component Developer 5.0
Sun Certified Mobile Application Developer

Wstęp

Harmonogram został opracowany w celu wstępnej prezentacji przebiegu szkolenia przygotowującego do egzaminu SCJP 6.0.

Na początku szkolenia, zostanie przedstawiona krótka charakterystyka samego egzaminu SCJP 1.6 (kwestie techniczne, rodzaje pytań, przebieg, etc.), po której rozpocznie się część merytoryczna. Szkolenie składa się w wielu zagadnień tematycznych, które wchodzą w skład egzaminu SCJP. Przebieg każdego działu będzie się składać z następujących elementów:

- wprowadzenie teoretyczne, poruszające wszystkie aspekty danego tematu
- zadania praktyczne do danego działu (każdy uczestnik będzie realizować je samodzielnie na własnym laptopie) w formie tworzenia prostych programów uwidaczniających sedno danego zagadnienia
- przedstawienie typowych problemów i pułapek jakie można spotkać na egzaminie, które to zostaną zademonstrowane na przykładowych pytaniach testowych z danego zagadnienia

Szkolenie będzie prowadzone w sposób interaktywny. Przez cały czas prowadzący szkolenie pozostaje do dyspozycji uczestników i będzie odpowiadać na ich ewentualne pytania oraz pomagać w rozwiązywaniu problemów. W harmonogramie nie wykazano przerw w trakcie szkolenia – aczkolwiek zostawiono rezerwę czasową na nie przeznaczoną (ok. 40 [min] w ciągu 8 [h] co daje średnio ok. 5 [min/h]). Pod koniec każdego dnia przewidziano:

- krótkie podsumowanie omówionego materiału
- powtórzenie najistotniejszych punktów
- przedstawienie przykładowych pytań do samodzielnego rozwiązania przez uczestników szkolenia.
- sprawdzenie wyniku oraz dyskusja w celu wyjaśnienia ewentualnych wątpliwości

Harmonogram

Dzień 1

Podstawy i elementy języka

Teoria	Ćwiczenia praktyczne	Czas [min]
Egzamin SCJP – charakterystyka i przebieg	pytania i odpowiedzi	20
użycie modyfikatorów dostępu, deklaracje pakietów, polecenia importu	przykład importu statycznego, stosowanie zamiennie importu lub pełnej nazwy kwalifikowanej, demonstracja zakresu widoczności dla różnych modyfikatorów dostępu, domyślne zakresy widoczności	30
uruchamianie klas z linii poleceń	demonstracja użycia java.exe z plikami class oraz JAR, parametry java.exe, przekazywanie parametrów do aplikacji, parsowanie parametrów liczbowych	40
różnica w zachowaniu się obiektów i typów prymitywnych podczas przekazywania jako argumenty metod	stworzenie metody przyjmującej obiekt i prymityw – ukazanie różnic, deklaracje final, prosty sposób przekazywania prymitywów przez referencje	40
określanie miejsca w którym obiekt jest przeznaczony do usunięcia przez GC; zachowanie GC (co jest gwarantowane, a co nie), użycie metody Object.finalize()	demonstracja działania GC poprzez pokrycie metody finalize (wypisanie komunikatu na ekranie), efekty wywołania metody Runtime.gc	30
określanie prawidłowej struktury katalogów w archiwum JAR na podstawie pełnej nazwy klasy; określanie wymaganej wartości CLASSPATH do prawidłowej kompilacji danej klasy	kilka przykładów użycia CLASSPATH dla plików class oraz JAR, użycie komendy JAR do tworzenia archiwum, wykorzystanie stworzonego archiwum	30
użycie operatorów przypisania (=, +=, -=), arytmetycznych (+, -, *, /, %, ++, --), porównania (>, >=, <, <=, ==, !=), operator instanceof, operatory logiczne (&, |, ^, !, &&, ||), operator warunkowy (? :), porównywanie obiektów oraz typów prymitywnych	działanie operatora %, instanceof, różnice pomiędzy operatorami (& i && oraz | i || - pełna i skrócona ewaluacja), użycie operatora warunkowego, różnice w porównywaniu obiektów i prymitywów	40
deklaracje klas, interfejsów, klas zagnieżdżonych, typów wyliczeniowych, deklaracje i import pakietów, import statyczny	zadeklarowanie klasy, interfejsu, klasy zagnieżdżonej, użycie importu statycznego, wykorzystanie typu enum i zaprezentowanie jego zalet w porównaniu do starego sposobu implementacji (public static final int)	40
implementacja interfejsów, deklaracje i dziedziczenie klas abstrakcyjnych	stworzenie prostej hierarchii kilku klas wraz z klasa abstrakcyjna i interfejsem	40

deklaracje, użycie i inicjalizacja zmiennych prymitywnych, tablicowych, wyliczeniowych i obiektowych; atrybuty statyczne, instancji oraz lokalne, dopuszczalne nazwy identyfikatorów	stworzenie klasy z polami statycznymi, instancji, tablicami wraz z inicjalizacją, kompilacja klasy z nieprawidłowymi identyfikatorami, inicjalizacja statyczna klasy	**40**
deklaracje metod statycznych i niestatycznych, konwencja nazewnicza dla metod JavaBeans, użycie metod o zmiennej liście parametrów	stworzenie i użycie prostej klasy JavaBean (np. Person), stworzenie i użycie metody o zmiennej liczbie parametrów	**30**
Przykładowe pytania, dyskusja		**60**
	Suma	**440**

Dzień 2

Elementy języka i sterowanie aplikacją

Teoria	Ćwiczenia praktyczne	Czas [min]
pokrywanie i przeciążanie metod	hierarchia kilku klas demonstrująca polimorfizm i przeciążanie	50
tworzenie konstruktorów, konstruktory domyślne, tworzenie instancji klas (zwykłych, zagnieżdżonych i anonimowych)	użycie klas zagnieżdżonych, ich specyfika i ograniczenia, parametry metod w klasach zagnieżdżonych, zasady tworzenia i użycie klas anonimowych na – stworzenie prostego Shootdown hook-a w oparciu o klase Thread	70
blok wyboru (switch) oraz jego użycie	prosty metoda sterowana blokiem switch, użycie klauzuli default, uzyskiwanie różnorakich efektów używając słowa kluczowego break, specyficzne użycie switch w połączeniu z typem enum	40
użycie i formy pętli (for, while, do, for-each), iteratory, etykiety petli, sterowanie petlami (break, continue), liczniki w petlach	stworzenie petli zagnieżdżonych z wykorzystaniem etykiet i instrukcji break, continue	50
zastosowanie i użycie asercji	prosty przykład używający asercji w celu walidacji warunków wykonania, kompilacja z włączeniem/wyłączeniem asrcji	20
obsługa wyjątków, bloki: try-catch, try-finally, zasady użycia wyjątków przy pokrywaniu metod	konstrukcja bloków try, catch, finally, prosta hierarchia klas demonstrująca użycie wyjątków przy pokrywaniu metod, użycie polecenia return w bloku try	55
skutki wystąpienia wyjątków w określonych blokach kodu, rodzaje wyjątków (runtime, checked, error)	przykład skutku wystąpienia wyjątku w kodzie (konstruktor, metoda, inicjalizer, bloki try, catch, finally)	25
sytuacje w jakich wyrzucane są wyjątki: ArrayIndexOutOfBoundsException, ClassCastException, IllegalArgumentException, IllegalStateException, NullPointerException, NumberFormatException, AssertionError, ExceptionInInitializerError, StackOverflowError, NoClassDefFoundError	sytuacje z przykładami kodu wyrzucającymi wszystkie w/w wyjątki	50
podział wyjątków: ze względu na wyrzucane przez JVM oraz wyrzucane w kodzie		20
Przykładowe pytania, dyskusja		60
	Suma	440

Dzień 3

Podstawy API

Teoria	Ćwiczenia praktyczne	Czas [min]
wrappery klas dla typów prymitywnych (Character, Byte, Integer, etc.), użycie autoboxing-u oraz unboxing-u; różnice pomiędzy String, StringBuffer i StringBuilder	przykład użycia wrapperów z autoboxingiem i unboxingiem, demonstracja różnic pomiędzy String, StringBuilder, StringBuffer, specyfika zachowania klasy String (operator == i metoda equals)	60
operacje na systemie plików, odczyt i zapis plików, interacja z użytkownikiem, użycie następujących klas z pakietu java.io: BufferedReader, BufferedWriter, File, FileReader, FileWriter, PrintWriter, Console	program zapisujący tekst wpisany z konsoli do pliku tekstowego, użycie klasy Console, wykorzystanie PrintWriter do prezentacji typów prymitywnych	60
serializacja i deserializacja obiektów z użyciem pakietu java.io: DataInputStream, DataOutputStream, FileInputStream, FileOutputStream, ObjectInputStream, ObjectOutputStream, Serializable	Stworzenie prostej klasy implementującej Serializable i serializacja do pliku, własna implementacja serializacji	70
użycie pakietu java.text do formatowania i parsowania dat, liczb i walut wraz ze specyficzną lokalizacją (Locale), użycie domyślnej lub wskazanej lokalizacji, zastosowanie klasy java.util.Locale	formatowanie dat i walut dla różnych lokalizacji, parsowanie dat, czasu z użyciem DateFormat	60
zastosowanie pakietu java.util do formatowanie i parsowania łańcuchów oraz strumieni, użycie wyrażeń regularnych, klasy Pattern, Matcher, metoda String.split, wzorce wyrażeń regularnych *, +, ?, \d, \s, \w, [], (); użycie klas Formatter, Scanner oraz metod klasy PrintWriter: format i printf, parametry formatujące (%b, %c, %d, %f, %s)	zastosowanie wyrażeń regularnych do walidacji adresu email, numeru NIP, nr rachunku bankowego, znajdowanie i zastępowanie tekstu z użyciem wyrażeń reg., użycie klasy Scanner do czytania wartości z konsoli – prosty program czytający różne typy danych z konsoli, użycie metody printf do formatowania podstawowych typów danych – działanie parametrów formatujących	130
Przykładowe pytania, dyskusja		60
	Suma	440

Dzień 4

Konkurencja i obiektowość

Teoria	Ćwiczenia praktyczne	Czas [min]
definiowanie, tworzenie i uruchamianie wątków z użyciem klasy Thread i interfejsu Runnable	utworzenie prostego wątku wykonującego długotrwałe obliczenia, nie blokującego interfejsu użytkownika	35
stany wątku i przejścia między stanami	startowanie, przerywanie, usypianie, oczekiwanie, blokowanie, użycie yield	25
blokowanie zasobów (statycznych i instancji) przy dostępie konkurencyjnym	przykład konkurencji bez blokowania i z blokowaniem, użycie słowa synchronized na obiekcie oraz w deklaracji metody	40
wykorzystanie metod wait, notify, notifyAll	demonstracja działania wait, notify; różnice pomiędzy notify i notifyAll, przykład typu producer-consumer	70
zagadnienia hermetyzacji (encapsulation), luźnego powiązania (loose coupling) oraz spójności (cohension) wraz z korzyściami jakie zapewniają	stworzenie klasy zgodnie z zasadami OO, demonstracja ścisłego powiązania (tight coupling) i luźnego powiązania (loose coupling) oraz wysokiej i niskiej spójności (high, low cohension)	50
użycie polimorfizmu, zastosowanie rzutowania typów i warunki jego użycia, błędy rzutowania na etapie kompilacji (compile-time) oraz uruchamiania (run-time)	hierarchia kilku klas z użyciem polimorfizmu, przykład prawidłowego i nieprawidłowego rzutowania	40
wykorzystanie modyfikatorów w dziedziczeniu, dla konstruktorów, pól (statycznych i instancji), oraz metod (statycznych i instancji)	demonstracja działania modyfikatorów dostępu, realizacja Singleton-u, stworzenie prostej fabryki klas	45
wywoływanie metod pokrywanych oraz przeciążonych, deklaracja i wywołanie konstruktorów przeciążonych oraz konstruktorów z klasy bazowej	demonstracja na hierarchii klas, różnica pomiędzy przeciążaniem a pokrywaniem, wywołanie konstruktora z tej samej klasy, z klasy bazowej	45
implementacja relacji jest (is-a, is-like-a) oraz posiada (has-a)	kilka klas demonstrujących w/w relacje (Car, Truck, Engine)	30
Przykładowe pytania, dyskusja		60
	Suma	440

Dzień 5

Kolekcje i generyki

Teoria	Ćwiczenia praktyczne	Czas [min]
wybór i zastosowanie poszczególnych klas kolekcji (listy, zbiory, mapy)	przykłady użycia klas, demonstracja różnic w funkcjonalności oraz wydajności	50
prawidłowe pokrywanie metod equals i hashCode, różnica pomiędzy wynikiem metody equals(), a operatora ==	demonstracja wpływu pokrycia equals oraz hashCode na zachowanie kolekcji	50
użycie kolekcji generycznych (list, map, zbiorów), ograniczenia kolekcji niegenerycznych, refaktoring kodu do użycia kolekcji generycznych, wykorzystanie interfejsów NavigableSet i NavigableMap	przykład użycia prostej kolekcji generycznej	60
użycie parametrów w deklaracjach klas oraz interfejsów, pól instancji, argumentach metod, wynikach metod; tworzenie metod generycznych oraz metod używających symboli wieloznacznych, różnice między nimi	stworzenie klasy generycznej z metodami generycznymi	50
operacje na listach udostępniane przez pakiet java.util: sortowanie, wyszukiwanie binarne, konwersja listy do tablicy	sortowanie i wyszukiwanie wartości w liście	40
operacje na tablicach (pakiet java.util): sortowanie, wyszukiwanie binarne, konwersja na liste	sortowanie i wyszukiwanie wartości w liście	40
użycie interfejsów java.util.Comparator i java.lang.Comparable, do uzyskania otrzymanego porządku sortowania; otrzymanie porządku naturalnego z użyciem klas wrapperów dla typów prymitywnych oraz klasy String	Stworzenie prostej klasy (Person) wraz z komparatorem; wykorzystanie jej w kolekcji, implementacja sortowania po imieniu, nazwisku oraz dacie urodzenia	60
Przykładowe pytania, dyskusja		90
	Suma	440

Uwagi

Niniejszy harmonogram ma charakter poglądowy i w trakcie szkolenia może ulec zmianie.