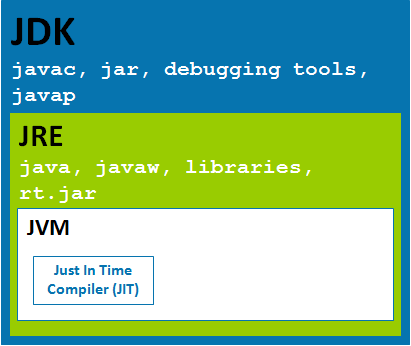
1. JRE vs JDK

Java Runtime Environment pozwala na uruchomienie aplikacji napisanej w języku Java. Java Development Kit umożliwia tworzenie aplikacji i zawiera w sobie również JRE.



1. JVM – Java Virtual Machine to wirtualny „komputer”, który tłumaczy kod napisany w Javie (zrozumiały dla programisty) na kod bajtowy zrozumiały dla urządzenia, na którym kod ten ma być uruchomiony.
2. Chcemy skompilować klasę Hello.java (konsola), jakiej użyjemy komendy?

javac Hello.java

1. Jakie są możliwe sygnatury metody main?

public static void main(String[] args) lub (String … args)

1. Pamięć w Javie - gdzie przechowywane są obiekty?

Na stercie

1. Kiedy obiekt jest nieużytkiem (Garbage Collector)?

Kiedy nie ma już dla niego żadnej referencji

1. public class Main {/\*…\*/} - jakie są możliwe nazwy pliku źródłowego tej klasy?

Main.java

1. Pole statyczne

Wspólne dla wszystkich obiektów danej klasy

1. Metoda statyczna

Można ją wywołać bez tworzenia obiektu danej klasy

1. import vs import static vs static import

static import nie działa

import static pozwala na użycie jakiegoś elementu bez podawania nazwy pakietu, z którego pochodzi, np. import static java.lang.Math.PI pozwala na pisanie PI zamiast Math.PI (tego używamy, gdy napiszemy tylko import java.lang.Math.PI)

1. do/while vs while/do

do/while zawsze wykona się co najmniej raz, bo najpierw działa polecenie, a dopiero później sprawdzany jest warunek. While/do najpierw sprawdza warunek, więc polecenie może się nie wykonać ani razu.

1. JavaBeans – przekazywanie jakichś stanów lub informacji za pomocą „ziaren” do innych elementów aplikacji.
2. Czy w tablicy możemy przechowywać elementy o różnych typach?

Nie, ale mogą to być obiekty klasy nadrzędnej, np. obiekty Vehicle, chociaż jeden z nich jest obiektem Car, a inny obiektem Plane, czyli są instancjami różnych klas rozszerzających Vehicle

1. switch(x) - jakie są dopuszczalne typy x?

byte, short, int, char, enum, String

1. protected vs domyślny modyfikator

default: nie jest widoczne w żadnej klasie z innego pakietu

protected: jest widoczne dla klasy pochodnej z innego pakietu

Zarówno default jak i protected są widoczne w swoim pakiecie

1. Ile klas publicznych można zdefiniować w jednym pliku?

jedną

1. Czy to poprawna nazwa identyfikatora : \_2323HDY$$dkcn ?

Poprawna, ale niezgodna z konwencją

1. int x = i++; vs int x = ++i;

w pierwszym przypadku do x zostanie przypisana wartość i, a następnie i zostanie powiększone o 1

w drugim przypadku i zostanie powiększone o 1 i przypisane do x, więc i oraz x będą sobie równe

1. Zwartość (cohesion)

Klasa powinna wykonywać jedno konkretne zadanie. Dąży się do jak największej zwartości.

1. Spójność (coupling)

Klasa powinna łączyć się z jak najmniejszą liczbą innych klas, by w momencie wprowadzenia do niej zmian nie zepsuć działania w innych klasach. Dąży się do jak najmniejszego sprzężenia.

1. Klasa vs obiekt

Klasa to szablon, na podstawie którego tworzone są obiekty. Klasa opisuje, jakie cechy ma obiekt i jakie funkcje potrafi wykonywać. Obiekt jest instancją klasy, czyli elementem zbudowanym na podstawie szkieletu i ma swoje konkretne cechy przypisane do pól i wykonuje zadania opisane przez metody.

1. Hermetyzacja (encapsulation)

Ukrywanie danych o obiekcie, by nie można było wprowadzić do nich zmian. Np. obiekt może mieć pole prywatne, które możemy odczytać lub zmienić za pomocą publicznych getterów i setterów.

1. Agregacja vs kompozycja

Agregacja to relacja, w której jeden obiekt „zawiera” inne obiekty, które mogą istnieć bez niego, np. obiekt szkoła może zawierać obiekty typu uczeń.

Kompozycja to relacja, w której jeden obiekt „posiada” obiekt innego typu i ten „posiadany” obiekt nie może istnieć sam. Czas życia obiektu komponowanego nie przekracza czasu życia obiektu głównego. Np. obiekt osoba „posiada” obiekt pesel. Pesel jest tworzony w momencie tworzenia osoby i nie może istnieć sam, bez osoby.

1. Polimorfizm

Obiekt przechodzi przez wiele testów istanceof. Można tworzyć wiele referencji do jednego obiektu i na ich podstawie modyfikować dostęp do metod. Każdy obiekt jest polimorficzny, bo dziedziczy co najmniej po klasie Object.

1. Czy możemy zdefiniować prywatną klasę?

Możemy wewnątrz klasy, która nie jest private.

1. Czy możemy zdefiniować prywatny interfejs?

Tak

1. Do czego służy adnotacja @Override

Do nadpisania metody. Adnotacja ta sprawdza, czy spełnione są warunki nadpisania.

1. Jak wywołać konstruktor klasy bazowej?

super()

1. Czy konstruktor może wywoływać przedefiniowane metody?

nie

1. Kontrakt equals & hashCode

Jeśli obiekty są takie same według equals, to będą miały takie same hashcody. Jak mają takie same hashcody, to nie muszą być equal, bo hashcode nie jest unikalny.

1. Jakie jest domyślne zachowanie metody toString ?

Wypisanie kodu obiektu

1. Jak porównujemy typy String ?

Za pomocą equals

1. W jaki sposób możemy tworzyć obiekty typu String?

String s = „siema”; <- w String pool

String s = new String(„siema”); <- w pamięci lokalnej

1. Czy możemy rozszerzać klasę Integer?

Nie, bo to klasa final

1. Różnica StringBuffer vs StringBuilder, który najlepiej stosować?

StringBuffer jest kompatybilny wielowątkowo, a Builder nie

1. Do czego używamy final?

Do tworzenia stałych lub klas, których nie można rozszerzać

1. Programowanie do interfejsów

Trzeba zaimplementować metody zdefiniowane w interfejsie

1. Do czego możemy użyć statycznej metody w interfejsie?
2. Mamy klasy:

abstract class Vehicle

abstract class Ship

class BattleShip extends Ship

W klasie Vehicle mamy metodę abstract void start();

Która klasa w hierarchii dziedziczenie musi zaimplementować metodą start()?

1. UML - jak oznaczamy statyczne elementy klasy?

kursywą