1.A. W jakim celu używa się klas abstrakcyjnych, a w jakim interfejsów?

Zarówno klasy abstrakcyjne, jak i interfejsy służą do tworzenia dodatkowej warstwy abstrakcji w aplikacji. Oba typy pozwalają na stosowanie bardziej uogólnionego kodu i unikanie powtórzeń, umożliwiają również polimorfizm.

Odpowiedź oprę o Javę 8 i wyższe wersje, z uwagi na pojawienie się w Javie 8 interfejsów funkcyjnych, co spowodowało, że interfejsy zyskały na znaczeniu.

Interfejsy

Ogólnie rzecz biorąc, interfejs definiuje zachowanie, które powinny ‘mieć’ klasy go implementujące.
Interfejsy ogrywają dużą rolę w stosowaniu zasad programowania obiektowego i wzorców projektowych. Między innymi pozwalają realizować w praktyce Open/closed - , Liskov sustitution -, Dependency inversion principle. Umożliwiają hermetyzację kodu programu, który jest podatny na zmiany, pozwalają stosować kompozycję w miejsce dziedziczenia i dają możliwość tworzenia luźnych powiązań miedzy obiektami, bez przywiązywania się do konkretnych implementacji.

Co istotne, jedna klasa może implementować kilka interfejsów naraz.

Jeśli chodzi o wspomniane interfejsy funkcyjne, to są to interfejsy, które posiadają tylko jedną metodę abstrakcyjną i umożliwiają realizację paradygmatu programowania funkcyjnego. Do zmiennej typu interfejsu funkcyjnego można przypisać wyrażenie lambda, referencję do metody lub konstruktora i znacznie skrócić zapis kodu, eliminując konieczność tworzenia w tym miejscu klasy anonimowej. Pozwala to przede wszystkim na większą ekspresywność kodu.

Klasy abstrakcyjne

Klasy abstrakcyjne służą przede wszystkim do definiowania typu bazowego w obiektach modelu danych. Możemy zdefiniować typ nadrzędny, po którym mogę dziedziczyć klasy będące jej podtypem. Dla przykładu, możemy zdefiniować abstrakcyjną klasę Vehicle, po której będą dziedziczyć klasy Motorbike i Car. Klasa abstrakcyjna, w przeciwieństwie do interfejsu, może przechowywać stan obiektu – posiadać pola instancyjne, dzięki czemu możemy zawrzeć w niej własności wspólne dla wszystkich klas dziedziczących.
Korzystanie z klas abstrakcyjnych pozwala przede wszystkim na redukcję powtórzeń w kodzie oraz stosowanie polimorfizmu.

1.B. Czym różni się tablica od listy liniowej?

Tablica to podstawowa struktura danych, w której można przechowywać elementy w uporządkowanej formie. Tablice mogą być jednowymiarowe, jak i wielowymiarowe. Struktura tablicy jednowymiarowej przypomina ponumerowany szereg, a np. w przypadku tablicy dwuwymiarowej macierz. Tablice charakteryzują się stałą wielkością, a odwoływanie się do poszczególnych elementów w tablicy odbywa się poprzez odwołanie do konkretnych indeksów.

Listy liniowe są, w przeciwieństwie do tablic, strukturami o dynamicznym rozmiarze, który może ulegać zmianie w czasie działanie programu. Implementacja listy liniowej opiera się na ‘węzłach’, czyli obiektach, które pozwalają na przechowywanie danych i dodatkowo posiadają referencje na sąsiadujące elementy – jeśli mamy do czynienia z listą jednokierunkową, węzeł posiada odniesienie

do następnego elementu, jeśli jest to lista dwukierunkowa, węzeł posiada również informację o poprzednim elemencie.
Lista liniowa jest strukturą, w której kolejność elementów ma znaczenie, a każdy z elementów posiada swój indeks.

Sposób przechowywania elementów w pamięci w przypadku tablicy i listy liniowej jest różny. Tablica potrzebuje jednolitego obszaru w pamięci komputera, a lista wiązana przetrzymuje elementy rozrzucone w pamięci.

Dodatkowo w przypadku obu struktur, mamy inne złożoności obliczeniowe dla operacji pobierania, dodawania czy usuwania elementów.