Podstawowe Komendy:

Psvm – skrót klawiszowy do stworzenia klasy głównej (main)

System.out.print(„napis”) – wydrukuj na ekranie

System.out.println(„napis”) – z nową linią

Sout – skrót klawiszowy

Var zmienna = … - inteligentna zmienna

Scanner scan = new Scanner(System.in) – deklaracja skanera

Int liczba = scan.nextInt() – wczytanie liczby od użytkownika

String napis = scan.next() – wczytanie napisu od użytkownika

Int[] tablica = {1, 2, 3} – tablica liczb z określonym z góry rozmiarem

System.out.println(Arrays.toString(tablica)); - wyświetla tablice

Int[][] tablica = {{}, {}, {}} – tablica dwuwymiarowa

System.out.println(Arrays.deepToString(tablica)); - wyświetl tablice dwuwymiarową

Try{ „Kod” } – Zaznaczasz, że w tym miejscu może pojawić się błąd

Catch (wyjątek) {„kod”} – w razie problemu zaznaczonego w wyjątku zrób …

Finally { „kod” } – to się zawsze wykona po wystapieniu lub nie wystąpieniu wyjątku

Throw new wyjątek – zwróć wyjątek

Obiektowość:

Static – Wspólna zmienna dla wszystkich obiektów, wspólna funkcja dla wszystkich klas

Final – Zmiennej/Funkcji nie da się zmienić (const)

Public – Zmienną/Funkcję można zmieniać/wyświetlać z poziomu całego kodu

Private – Zmienną/Funkcję można zmieniać/wyświetlać tylko z poziomu klasy w której ona powstała

Protected – Private, ale z dostępem z poziomu klas dziedziczących

Constructor – funkcja uzupełniająca zmienne na starcie klasy

Setter – funkcja pozwalająca zmienić zmienną prywatną

Getter – funkcja pozwalająca wyświetlić zmienną prywatną

Przeciążanie – tworzenie 2 takich samych funkcji z innymi typami zmiennych

Kolekcje – listy, stosy, kolejki itd.

Public interface Nazwa {funkcja} – wzór zawierający funkcje tylko do dziedziczenia

Public class Naz implements nazwa – dziedziczenie po interfejsie

Nazwa obiekt = new Naz(); - tworzenie obiektu (Najpierw ogół-interfejs, potem klasa dziedzicząca)

Public abstract class Nazwa {…} – interfejs, ale może przechowywać również deklaracje zmiennych

Public class Naz extend Nazwa – dziedziczenie po abstrakcji

@Override – Nadpisanie wydziedziczonej zmiennej/funkcji

Typy Generyczne:

Rzutowanie – zmiana typu

Typ Generyczny – zapis pozwalający uniknąć rzutowania

List<String> = new ArrayList<String>() – Lista przechowująca Stringi

List<Object> = new ArrayList<>() – Lista przechowująca dowolne objekty

Public class Nazwa<T>{ … } – klasa generyczna

Class Nazwa<T, G> {…} – klasa generyczna z 2 możliwymi typami

T obiekt1; - obiekt z dynamicznym typem

Nazwa<Typ> naz = new Nazwa<>(); - wywołanie klasy

Nazwa<String> napis = new Nazwa<>(); - przykład

Class Nazwa<T extends String> {…} - Ograniczenie klas, Tylko String i jego dziedziczące

<?> - wildcard, czyli nieznany parametr. Takie T, ale okrojone

https://www.geeksforgeeks.org/wildcards-in-java/

Lambdy:

@FunctionalInterface – Interfejs posiadający tylko jedną funkcję

Interface nazwa = (zmienne) -> wnętrze; - Lambda. Skrócona wersja tego:

```
Interface nazwa = new Interface() {
        @Override
        Public int cokolwiek(zmienne) {
                Wnętrze;
        }
}
```

Np. public interface Dodawanie { int doda(int a, int b) } – tworzenie interfacu funkcj.

Dodawanie d1 = (a, b) -> a * b; - tworzenie lambdy

d1.doda(2, 4); - wykorzystanie lambdy

Strumienie:

Stream<Typ> nazwa = Stream.of(…) – tworzenie strumienia

Stream<String> alfabet = Stream.of(“a”, “b”, “c”) – przykład strumienia na Stringach

List<String> nazwa = Arrays.asList(“a”, “b”, “c”)

nazwa.stream() – przekształcenie listy na strumień

strumien.count() – zlicza ilość elementów strumienia

strumien.forEach(System.out::println) – wyświetl element/y strumienia

strumien.filter(n -> n % 2 == 0) – tylko elementy parzyste

strumien.filter(n -> n % 2 == 0).forEach(System.out::println) – wyświetl je

s -> Character.isUpperCase(s.charAt(0) – zaczynające się od wielkiej litery

s -> s.endsWith("a") – kończy się literką „a”

strumien.map(n -> n*3) – wykonaj operację na strumieniu

strumien.allmatch(n -> n%2==0) – Czy wszystkie elementy spełniają warunek? Tak/Nie

strumien.anymatch(n -> n%2==0) – Czy chociaż jeden element spełnia warunek? Tak/Nie

strumien.distinct() – pomiń powtórki

strumien.skip(2) – pomiń pierwsze 2 elementy

strumien.limit(2) – Wyświetl tylko pierwsze 2 elementy

strumien.sorted() – posortuj strumień

Operacja pośrednie (intermediate) – zwracające strumień

O. końcowe (terminal) – może zwrócić cokolwiek poza strumieniem

Refleksje:

Class – zawiera informacje o podanej klasie

Method – zawiera informacje o podanej metodzie

Constructor – zawiera informacje o podanym konstruktorze

Class<?> klasa = Class.forName(ObjectOne.class.getName()); - sprawdza nazwę klasy

Class<?> nazwa = obiekt.getClass(); - sprawdza nazwę klasy po obiekcie

Class<?> pakiet = Class.forName(ObjectOne.class.getPackageName()); - sprawdza nazwę pakietu

klasa.getDeclaredFields() – pola istniejące w klasie

klasa.getDeclaredMethods() – metody istniejące w klasie

Adnotacje:

Adnotacja – Informacja dla kompilatora o co nam chodzi

Retencja – informacja o tym jak długo element ma być przechowywany

@Target(…) – Dla jakich elementów ma działać nasza adnotacja

@Target(ElementType. LOCAL_VARIABLE) – ma działać dla zmiennych

@Target(ElementType.METHOD) – ma działać dla metod

@Target(ElementType.TYPE_PARAMETER) – deklaracja typu parametru (generyczne)

http://tomasz.kubik.staff.iiar.pwr.wroc.pl/dydaktyka/Java/JavaAnnotations2019.pdf

@Retention(…) - czy adnotacja ma istnieć podczas działania programu?

@Retention(RetentionPolicy.RUNTIME) – tak ma istnieć

@Retention(RetentionPolicy.CLASS) – nie muszą być

@Retention(RetentionPolicy.SOURCE) – nie są

Public @interface Nazwa{…} – Deklaracja nowej adnotacji o nazwie Nazwa

{ int min(); } – Zmienna adnotacyjna

{ int min() default 0; } – Defultowo wynosząca 0

Testy Jednostkowe:

@Test – etykieta testu

AssertEquals(oczekiwana_liczba, testowana_metoda) – sprawdza czy rezultaty są identyczne

AssertEquals(o, t, zaokrąglenie) – assertEquals dla double

AssertTrue(testowana_metoda) – sprawdza czy wychodzi prawda

AssertFalse(testowana_metoda) – sprawdza czy wychodzi fałsz

Definicje:

Jednostka – część kodu przeznaczona do testów

Stub – obiekt zastępczy zrobiony na potrzeby testu

Mock – klasa zastępcza zrobiona na potrzeby testu

Punkt brzegowy – punkt w którym kończy się warunek (n > 60 -> pkt brzegowy = 60)

Wartości brzegowe – punkty, które musimy sprawdzić (59, 60, 61)

Dziedzina – zbiór elementów akceptowanych przez funkcję testującą

Strefa – część zbioru elementów akceptowanych

Klasa Równoważności – dane wejściowe/wyjściowe dla których oczekujemy podobnych rezultatów

Analiza war. Brzegowych – testujesz skrajne punkty i w ten sposób odkrywasz klasę równow.

Pozostałe:

Import java.util.Random – biblioteka pozwalająca na losowe liczby

Random rand = new Random() – deklaracja liczb randomowych

Int liczba = rand.nextInt(początek, koniec) – tworzenie liczby losowej

LocalDateTime dzisiaj = LocalDateTime.now() – tworzy zmienną z dzisiejszą datą

Import java.time.LocalDateTime – wymagane

DateTimeFormatter format = DateTimeFormatter.ofPattern("format ") – tworzenie zmiennej z formatem daty

import java.time.format.DateTimeFormatter – wymaga

dd – dzień (2 znaki)

MM – miesiąc (2 znaki)

yyyy – rok (4 znaki)

hh – godzina (2 znaki)

mm – minuta (2 znaki)

ss – sekunda (2 znaki)

String sformatowanaData = data.format(format) – tworzenie sformatowanej daty