

POO (Java)

javac + arquivo.java: compila
java + arquivo.java: executa o .class
JVM (virtual machine): executa o bytecode em qualquer SO

ao executar o programa pelo terminal, você consegue passar argumentos, que são args do public static void main.

WORA: write once, run anywhere

JDK: development kit

Sigre o Camel Case

Tipos Primitivos

char é tratado como número ASCII
usa menor memória que a classe do tipo.

Java não é 100% orientado a objetos porque tem tipos primitivos

char → 1 Byte

String → 1 Byte p/ letra

byte → -128 a 127

short → -32.768 a 32.767

int → 2 bilhões 747

long → 2⁶³

float → 2³²

double → 2⁶⁴

Importante saber os limites

* Caso excede a capacidade de memória o menor valor do tipo

Operadores

& ^ - ou exclusivo

& & : curto circuito: se a primeira expressão for falsa, ele nem olha a segunda

& : analisa as duas expressões

! ; " "

Print Formatted

%1s: string | %c: caractere

%d: int

%f ou %lf: double

%f float

%b Boolean

POO

1 / 1

Forma diferente de programar que resolve os problemas do programação procedural

Vantagens:

Confável

Eportano: desenvolvido independente

Mantenível

Extensível

Reutilizável

Natural

modificadores de acesso:

+ public: todas as classes

- private: apenas própria classe

protected: classe filha independente do pacote

default: apenas o pacote

static: suas componentes da classe não precisam instanciá-la

Herança

"É um" extends

super: permite acesso aos componentes da superclasse

importante a visibilidade

Polimorfismo

anulado com métodos

Anotação do método: retorno ou parâmetros diferentes.

sobrecarga | sobrecarga

override anotação

Abstract:

classe: não pode ser instanciada

abstrato: não existe abstrato

método: apenas declaração

* Se uma classe abstrata estender outra abstrata, ela não precisa implementar apenas o 1º classe não abstrata

Encapsulamento

esconder detalhes do objeto e sua implementação.

Código seguro

Final:

classe: não pode ser estendida

atributo: valor constante

método: não pode ser sobreescrito

Pacotes

sempre com todas as letras minúsculas. Dominio + nome do projeto ao contrário

Interface

É um contrato com a classe

Se não tiver atributos e apenas métodos abstratos, a classe abstrata pode virar uma interface

Torna que o Java encontra para contornar o problema de não suportar herança múltipla

pode ter métodos estáticos com implementação

Todos os métodos são públicos abstratos

Podem extender interfaces e a primeira classe que implementá-la, implementa todos

Use: métodos em comum com implementação específica

Classe Abstrata x Interface

Pode ter outros modificadores, implementação parcial
Não precisa ser constantes

public abstract
nenhuma implementação
Se tiver variáveis, não constantes (public static final)

Podem ter construtores
Pode ter métodos estáticos

Não tem construtor
Métodos não podem ser estáticos

Importante saber bem a diferença

Exceções

Código após o erro não é executado, vai para o catch.

try com recursos: só pode ser usado com objetos que implementem `closeable` ou `autoCloseable`.

Ordem dos catch importa específico
→ genérico

~~throwable~~: supradas de exceções da mesma família podem ser colocadas em um mesmo catch catch (ex11 ex2 e) {}.

`System.exit(0)`: finaliza o programa inteiro.

Exceção x Erro

tratáxis não tratável

checked: o compilador verifica o erro unchecked: erro ocorre durante a execução, como o acesso a um índice errado.

`getMessage()`: descrição do erro
`printStackTrace()`:

`throw + exceção`: colocado na anotação do método para passar responsabilidade para quem for usar

Exceção própria: nova classe que estende `Exception`
com construtor ou não
sobrecreve o `getmessage()`

Enum

Maior controle para uso de constantes, sendo melhor que interfaces/classe

Não podem ser instanciados
Pode comparar com ==
Podem implementar interfaces

Variáveis

Só aceita parâmetros de classes.
É tratado como vetor

`long (int a, Integer... resto)`
parâmetro é novo poderia ser depois de parâmetro resto

Sealed classes

1 / 1

Definir quais classes ou interfaces
podem estender ou implementar outra

Palavra chave sealed antes de class
e definir quais classes podem estende-la
com permite...

Usado em projeto com escopo bem
definido

* JDK 16

Mas classes filhas, obviamente deve
dizer se é sealed, non-sealed a final

Records

Usado para guardar dados de um
objeto.

Já é implícito `toString`, `equals`,
`hashCode` e `get`.

Não tem método `set`

Não podem estender, mas podem implementar
métodos e atributos estáticos, normal

Strings

Declarar uma string com new
sempre terá um endereço de memória
diferente

Catenação: cria uma string
nova juntando os valores (não imutável),
Ruim para memória

Se for por atribuição simples, mas
tiver o mesmo conteúdo, vai apen-
sar para o mesmo endereço

Stringbuilder: não é threadsafe

Stringbuffer: threadsafe

* não vale a pena usar o buffer em
singlethread.

Comparar objetos

equals: deixar o automático ou implementar ao máximo

hashCode: deixar o java gerar o hashCode

* importante usarem os mesmos atributos

Comparable: apenas 1 por classe

Comparator: classe externa com a forma de ordenar

Usar lambda para ordenar.

Wrapper.compare(o1, o2)

Collections

Interface List

arraylist e linkedlist

Iterator: usado para fazer modificações durante um for

Interface Set

HashSet: organizados pelo hash (não implementado)

Não mantém a ordem de inserção eficiente para busca

LinkedHashSet: mantém a ordem

TreeSet: implementação baseada em árvores

NavigableSet: interface para implementar TreeSet (opção de ordenar o set)

* não sobrescreve elementos repetidos

Interface Map

dicionários. Permite uma chave nula e vários valores nulos

HashMap: não garante ordem

LinkedHashMap: mantém a ordem

TreeMap: não permite chaves nulas

Hashtable: implementação ligada, semelhante ao HashMap, mas sincronizada

ConcurrentHashMap: implementação para cenários com thread.

* Sobrepõe chave repetida

* Para ordenar transforma em um Array ou usa stream

Interface Queue

linked list ou array deque para fila comum

Priority Queue para prioridade