22-9-2022

1) Eclipse

JSE = Java Standard Edition - Java Developer ( Core Java)

JEE = Java Enterprise Edition - Web, services (Advanced Java)

2) JDK

3) Tạo Project

Name: group19

* exams
* objects

4) THuật toán kiểm tra số nguyên tố

Cho n chia hết cho các giá trị từ 2 -> phần nguyên dưới căn bậc 2 của n

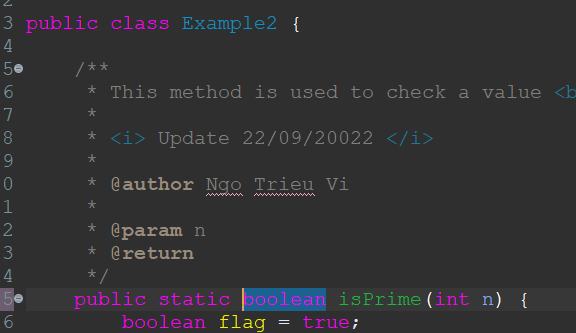
\*Note:

- Viết mã code theo khối

- Chú thích rõ ràng

- Sử dụng thẻ HTML để định dạng comment

- Gắn phương thức vào lớp đối tượng thông qua từ khóa “static”



- Overloading

BVN:

// Giải thuật

Tính tích phân có 2 cận

Tính thể tích hình trụ, hình nón

Tính tổ hợp và chỉnh hợp

Giải hệ 3 PT 3 ẩn

// Hướng đối tượng

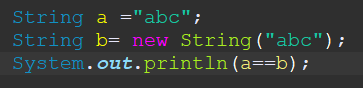
Class Address (cityName, districtName, streetName)

// Buổi 5

|  |  |
| --- | --- |
| Persistence Class | Business Class |
| Properties | Methods |

Mảng đối tượng: phần tử mảng lưu trữ địa chỉ của phần tử

Buoi 3:



The == operator compares references and not contents. It compares two objects and if they are the same object in memory and present in the same memory location, the expression returns true else it returns false. It is quite possible that two same contents are located in different memory locations.

BVN 20-10:

Str = “#abc####hmtn##nid#abc##loe##”;

=>

abc – 2

hmtn – 1

nid – 1

loe – 1

Tính tần suất xuất hiện của các từ.

Viết tên riêng chuẩn (viết hoa chữ cái đầu mỗi từ)

So sánh độ giống nhau của 2 chuỗi ký tự.

Cho chuỗi ký tự bất kỳ, hãy biến đổi định dạng của những chuỗi ký tự con giống chuỗi ký tự người dùng nhập.

Chuyển chuỗi tiếng Việt thành chuỗi thường.

Nạp chồng: chạy nhanh hơn, là các phương thức giống nhau về cấu trúc, khác nhau về tham số truyền vào, kiểu dữ liệu, xảy ra trong 1 lớp.

Ghi đè: xảy ra tại runtime, chạy chậm hơn, là một phương thức được thực hiện ở nhiều lớp cha con

Instanceof: thể hiện của .....

INTERFACE

Trong interface chỉ có PHƯƠNG THỨC TRỪU TƯỢNG và HẰNG

Comparable<T> so sánh đối tượng this với đối tượng khác compareTo()

Comparator<T> so sánh 2 đối tượng khác nhau compare()

BTVN: tạo lớp đối tượng product, tạo interface để quản lý các product con (số lượng product con >=3 vd: ô tô, xe máy, ... )

Với các sản phẩm có giá, sắp xếp tăng/giảm. Cung cấp các phương thức tìm kiếm theo tên, giá,... trong interface

**Collections**

Benefits of Java Collection

* Consistent and reusable APIs
* Less development time
* Performance
* Clean code

Interface in Collection

|  |  |
| --- | --- |
| Iterable interface | lặp trên các phần tử của tập hợp |
| Collection interface |  |
| List interface |  |
| Queue interface |  |
| Deque interface |  |
| Set interface |  |
| Sorted Set interface |  |
| Map interface |  |

Collections:

* Collection.synchronizedList

Vector:

* Tương tự ArrayList
* Hỗ trợ Synchronized
* Vector<T> v = new Vector<T>();

Queue:

* FIFO
* Khai báo và sử dụng với
* Queue<String> pq = new PriorityQueue<>();
* Queue<Integer> q = new LinkedList<>();
* ArrayBlockingQueue
* ArrayDeque
* java.util -> unbounded queues
* java.util.concurrent -> bounded queues

Set:

Map:

* Kiểu dữ liệu key-value
* Không trùng lặp
* Cài thông qua HashMap và TreeMap

Bài tập: Cho chuỗi ký tự dạng:

@@adc@@@@brk@@adc@oip@oip@brk@adc@@adc@qwe

Yêu cầu thống kê:

adc – 4

brk – 2

oip – 2

qwe – 1

Overrided methods:

* Instance method(s) belong to the Object of the class, not to the class i.e. they can be called after creating the Object of the class.
* Instance methods are not stored on a per-instance basis, even with virtual methods. They’re stored in a single memory location, and they only “know” which object they belong to because this pointer is passed when you call them.
* They can be overridden since they are resolved using **dynamic binding** at run time.

**Luồng vào ra tập tin:**

java.io (data streams)

java.nio (channels, buffers, selectors)

|  |  |  |
| --- | --- | --- |
|  | Input | Output |
| Byte streams | InputStream | OutputStream |
| Character streams | Reader | Writer |

* Là các lớp trừu tượng
* **read(), write()**
* Thay đổi vị trí con trỏ trong stream: **skip(), mark(), reset()**
* Đóng luồng **close()**

Kết nối luồng với bên gửi / bên nhận

|  |  |  |
| --- | --- | --- |
| Source / Receiver | Character streams | Byte streams |
| Memory | CharArrayReader,  CharArrayWriter | ByteArrayInputStream, ByteArrayOutputStream |
| StringReader, StringWriter | StringBufferInputStream,  StringBufferOutputStream |
| Pipe | PipedReader, PipedWriter | PipedInputStream, PipedOutputStream |
| File | FileReader, FileWriter | FileInputStream |

Xử lý luồng

|  |  |  |
| --- | --- | --- |
| Kiểu dữ liệu | Character streams | Byte streams |
| Buffering | CharArrayReader,  CharArrayWriter | ByteArrayInputStream, ByteArrayOutputStream |
| StringReader, StringWriter | StringBufferInputStream,  StringBufferOutputStream |
| Pipe | PipedReader, PipedWriter | PipedInputStream, PipedOutputStream |
| File | FileReader, FileWriter | FileInputStream |

* Phân cấp các lớp của luồng byte và luồng ký tự.

**Bộ đệm**

* Giảm được số lượng yêu cầu đọc ghi
* Đọc ghi file kích thước lớn, thay thế FileReader bằng BuferedReader

**Luồng nhị phân**

* Đọc, ghi thông qua DataInputStream, DataOutputStream
* Unicode text format

**Serialization**

**Pipe**

* Nhiều thread chạy song song -> muốn dữ liệu giao nhau
  + Sử dụng pipe

**File**

* File nhị phân
* File văn bản

BTVN: 1. viết chương trình tạo file .txt

Dòng 1: “Danh sách lớp”

Dòng 2: “STT “ + “Họ tên” +”Tuổi”

Dòng 3->n : stt, họ tên, tuổi

Dòng cuối : “TONG SO = ? Sinh vien”

Đọc file và hiển thị ra màn hình

2. file sắp xếp

3. file lọc tên, tuổi

Nên dùng **serializable**

Dữ liệu sinh ngẫu nhiên