Lê Đức Nghĩa-Module 2

Map & Tree

- HashMap:
 + Lưu trữ dữ liệu theo cặp key – value.
 + Có thể chứa tối đa 1 key null hoặc nhiều value null.
 + Không duy trì thứ tự sắp xếp.

- LinkedHashMap (giống HashMap nhưng khác 1 số đặc điểm):
 + Duy trì thứ tự thêm vào.

- TreeMap (giống LinkedHashMap nhưng khác 1 số đặc điểm):
 + Không chứa key null.
 + Phần tử đưa vào sẽ đc sắp xếp (mặc định sắp xếp tăng dần theo key).
 + Đối với kiểu dữ liệu do người dùng tự định nghĩa thì phải implement Comparable.

- Binary Tree:
 + Hoạt động dựa vào các node.
 + Mỗi node có tối đa 2 cây con (left/right sub-tree).
 + Node trên cùng là node gốc (root node), node không có con đc gọi là node lá (leaf node), các node có cùng cha là node anh em (sibling).
 + Node con bên trái có giá trị < node cha < node con bên phải

DSA-Stack,Queue

Generic – là cơ chế cho phép truyền dữ liệu vào như là tham số (tham số hóa kiểu dữ liệu),
cho phép sử dụng method, class, interface với nhiều kiểu dữ liệu khác nhau.
 + Không cần phải ép kiểu.
 + Xây dựng đc bài toán tổng quát, tái sử dụng đc mã nguồn.
 + Bắt đc lỗi lúc compile (biên dịch).
 + Một số quy tắt đặt tên:
  - E – element.
  - T – type.
  - N – number.
  - K – key.
  - V – value.

- Stack (ngăn xếp) – là cấu trúc dữ liệu dạng danh sách, thêm và xóa phần tử theo cơ chế FILO.

- Queue (hàng đợi) – là cấu trúc dữ liệu dạng danh sách, thêm và xóa phần tử theo cơ chế FIFO.

DSA-List

- Collection Framework – là 1 khuôn khổ cung cấp 1 kiến trúc để lưu trữ và thao tác tới nhóm các đối tượng. Tất cả các hoạt động mà bạn thực hiện trên 1 dữ liệu (tìm kiếm, phân loại, chèn, xóa,..) có thể đc thực hiện bởi Java Collection.

+ List (danh sách) – là cấu trúc dữ liệu tuyến tính, trong đó các phần tử được sắp xếp theo 1 thứ tự xác định, cho phép các phần tử được trùng lặp nhau.
 + Set (tập hợp) – là cấu trúc dữ liệu tuyến tính, trong đó mỗi phần tử chỉ xuất hiện duy nhất 1 lần (tương tự như tập hợp trong toán học), và tập hợp chưa đc sắp xếp (chưa xét đến các lớp triển khai).
 + Queue (hàng đợi) – là kiểu dữ liệu nổi tiếng với kiểu vào ra FIFO (first-in-first-out/ vào trước ra trước).

- Set và các lớp triển khai:
 + HashSet – các phần tử đc lưu trữ dưới dạng bảng băm, không duy trì thứ tự chèn, không lưu phần tử trùng lặp.
 + Linked HashSet – các phần tử đc lưu trữ dưới dạng bảng băm với cấu trúc dữ liệu dạng danh sách liên kết, duy trì thứ tự chèn.
 + Tree Set – sử dụng 1 tree cho lưu giữ, mặc định các phần tử đc sắp xếp tăng dần.

- Array List – là 1 class dạng list, implement từ mảng có thể thay đổi đc kích thước.

- Linked List – là 1 class dạng list, lưu trữ data theo cấu trúc danh sách liên kết (đơn, đơn vòng, đối, đôi vòng).

- Comparable chỉ sắp xếp theo 1 tiêu chí.
 => Class phải implements comparable<E>.
 => Override phương thức compareTo( ).

- Comparator sắp xếp theo 1 hoặc nhiều tiêu chí (vd: sắp xếp theo tên, điểm, id,...).
 => Class phải implements comparator<E>.
 => Override phương thức compare( ).

Clean Code

- Clean code – là thuật ngữ mô tả những mã nguồn “tốt”.
+ Đơn giản.
+ Dễ hiểu.
+ Dễ nâng cấp/ chỉnh sửa.
+ Định danh (tên biến, method, class, module, project,…) có ý nghĩa.
+ Ít sự phụ thuộc.
+ Phương thức phải thực hiện đúng chức năng, body không đc quá dài.
+ Thể hiện đc ý nghĩa của chương trình.
+ Phương thức phải thực hiện đúng chức năng, body không đc quá dài.
+ Thể hiện đc ý nghĩa của chương trình.

Clean Code

- Code smell – là thuật ngữ mô tả những đoạn code “xấu”.
 + Tên phương thức quá dài.
 + Phương thức thực hiện quá nhiều chức năng.
 + Phương thức có quá nhiều tham số.
 + Class có quá nhiều line.
 + Chứa những mã code trùng lặp.
 + Lạm dụng comment.

Code Smell

IO: Binary file & Serialization

- Serialization – là cơ chế cho phép chuyển đổi từ đối tượng (object) sang Byte Stream, ngược lại Deserialization là
cơ chế chuyển từ Byte Stream sang Object.
 + Trong Java, để 1 Object có thể Serialization thì class đó phải implement Interface Serializable.
 + Khi lớp cha đã implement Serializable thì lớp con không cần implement lại Serializable.
 + Từ khóa transient giúp cho thuộc tính không bị Serializable.

Search Algorithm

- Linear Search:
 + Duyệt qua tất cả các phần tử ở trong mảng, tìm thấy thì trả về index, không tìm thấy thì trả về -1.
 + Sử dụng trong trường hợp: mảng nhỏ, không sắp xếp.
 + Độ phức tạp của thuật toán: tốt nhất là O(1), xấu nhất là O(n).

Binary Search:
 + Mảng đầu vào phải đc sắp xếp tăng giảm.
 + Bước 1: kiểm tra phần cần tìm với phần tử giữa mảng.
 Bước 2: nếu phần tử cần tìm = = phần tử giữa mảng => trả về index.
 Bước 3: nếu phần tử cần tìm < phần tử giữa mảng => xét mảng con bên trái.
 Bước 4: nếu phần tử cần tìm > phần tử giữa mảng => xét mảng con bên phải.
 Bước 5: lặp lại Bước 1 -> Bước 4.
 + Độ phức tạp của thuật toán: tốt nhất là O(1), xấu nhất là O(log2(n)).

- Độ phức tạp của thuật toán: được hiểu là số phép toán thực hiện, phụ thuộc vào tập kết quả đầu vào n.
 + n đại diện cho số phần tử của mảng trong thuật toán sắp xếp, hoặc tìm kiếm.
 + n đại diện cho độ lớn trong trường hợp kiểm tra xem n có phải là số nguyên tố hay không

Sort Algorithm

- Bubble Sort: + Phát hiện đầu vào đã đc sắp xếp.
 + So sánh 2 phần tử liền kề, hoán vị 2 phần tử đó nếu không thỏa mãn điều kiện đang xét.
 + Độ phức tạp: tốt nhất là O(n), xấu nhất là O(n2).
 + Phát hiện đầu vào đã đc sắp xếp.

- Selection Sort:
 + Tìm phần tử bé nhất trong mảng, sau đó hoán vị phần tử bé nhất cho số đầu tiên của mảng, xét tiếp mảng từ vị trí 1 -> length-1, tiếp tục cho đến khi mảng đã ddc sắp xếp.
 + Độ phức tạp: tốt nhất/ xấu nhất đều là O(n2).

- Insertion Sort:
 + Chèn phần tử vào mảng con đã đc sắp xếp, đảm bảo thứ tự sắp xếp sau khi chèn.
 + Độ phức tạp: tốt nhất là O(n), xấu nhất là O(n2).

Sort Algorithm

- Exception – là 1 sự kiện bất thường, nó làm phá vỡ flow (luồng thực thi) của chương trình và có
thể làm chết chương trình.
 + Class ở mức cao nhất là Throwable.
 + Hai class con trực tiếp là Error và Exception.

- Checked Exception – xảy ra tại thời điểm compile (có thể gọi là compile time exceptions), bắt buột phải xử lí (handle).
 + FileNotFoundException (một file được chỉ định với đường dẫn không tồn tại, hoặc không có quyền truy cập vào file đó).
 + IOException (lỗi không đọc được file).
 + ClassNotFoundException (nếu không thể chuyển kiểu object này sang kiểu object khác hoặc không tìm thấy class muốn tham chiếu tới).
 + DatabaseException.
 + NoSuchFieldException.

- Cách xử lý: 2 cách:
 + throw/throws – ném ngoại lệ cho phương thức khác xử lí.
 + try – catch: xử lí ngay tại nơi xảy ra exception.

- Error – lỗi do môi trường thực thi (JVM), không thể handle + chương trình sẽ chết. Lớp Error định nghĩa các ngoại lệ mà không thể bắt (catch) từ
chương trình.
 + StackOverflowError (tràn vùng nhớ stack, stack đạt đến giới hạn tối đa).
 + VirtualMachineError (được ném ra khi máy ảo Java gặp lỗi nội bộ hoặc giới hạn tài nguyên, điều này khiến máy không thể hoạt động. Đây là
một cơ chế tự bảo vệ được JVM sử dụng để ngăn toàn bộ ứng dụng bị treo).
 + OutOfMemoryError (khi hết bộ nhớ trong heap, chủ yếu khi bạn cố gắng tạo một object và không có đủ không gian trên heap để phân bổ
đối tượng đó). Topic

- Unchecked Exception – xảy ra lúc runtime, không chắc chắn xảy ra, không bắt buộc phải handle tại thời điểm compile.
+ NullPointerException (được ném ra khi chương trình cố gắng tham chiếu tới một đối tượng nhưng nó không có vị trí nào trên bộ nhớ, tức là có giá trị null).
+ ArrayIndexOutOfBoundsException (truy nhập mảng với chỉ số không hợp lệ).
+ ArithmeticException (xảy ra khi chia cho số 0).
***+ NumberFormatException (xảy ra khi cố gắng để chuyển đổi một chuỗi thành một số, xuất hiện khi khởi tạo biến bằng Integer.parseInt(scanner.nextLine()) hoặc
Double.parseDouble(scanner.nextLine())).***
+ InputMismatchException (xảy ra khi cố gắng để chuyển đổi một chuỗi thành một số, xuất hiện khi khởi tạo biến bằng scanner. .nextInt() hoặc scanner. .next Double ()).

IO: Text File

- Stream – là hoạt động nhập xuất dữ liệu. Có 2 loại:
 + Character Stream – hỗ trợ nhập xuất dữ liệu cho Unicode.
 + Byte Stream – hỗ trợ nhập xuất dữ liệu cho Byte (nhập xuất theo nhị phân)

String & Regex

- String – là 1 lớp đc định nghĩa sẵn trong Java, sử dụng để lưu trữ/ làm việc với chuỗi trong Java.
 + Có 2 cách để gán biến String:
  - Gán trực tiếp cho biến (giống kiểu dữ liệu nguyên thủy).
  - Thông qua từ khóa “new”.
  => String là kiểu dữ liệu đặc biệt, có tính chất vừa nguyên th ủy, vừa đối tượng.
 + String là không thể thay đổi, bất biến (immutable).
 + Một số phương thức hay dùng: concat, length, subString, split, trim, charAt,..

- StringBuilder/StringBuffer – mô tả các dữ liệu dạng chuỗi có thể sửa đổi linh động (mutable).
 + Khác về đồng bộ:
  - StringBuffer: hỗ trợ đồng bộ (synchronization).
  - StringBuilder: không hỗ trợ đồng bộ.
 + Một số phương thức hay dùng: append (đây là phương thức quan trọng nhất), insert, delete,
 reverse, toString,…

- Regular Expression (Regex) – là 1 chuỗi được sử dụng để truy định dạng thức của 1 chuỗi khác,
đc sử dụng cho validata (xác thực) dữ liệu và tìm kiếm.

SOLID

- SOLID – là 1 trong những nguyên tắc (chỉ dẫn) để giúp chúng ta xây dựng đc các ứng dụng OOP hiệu quả, giúp lập trình
viên viết ra những đoạn code dễ đọc, dễ hiểu, dễ maintain (bảo trì).

- S – Single responsibility principle – Nguyên lý Trách nhiệm Duy nhất

- O – Open closed principle – Nguyên lý đóng mở.

- L – Liskov substitution principle – Nguyên lý thay thế Liskov.

- I – Interface segregation principle – Nguyên lý phân tách Interface.

- D – Dependency inversion principle – Nguyên lý đảo ngược phụ thuộc.