# Object và Class

* Biến local: Biến được định nghĩa bên trong method, constructor hoặc blocks gọi là biến local. Biến sẽ được khai báo và khởi tạo trong method và sẽ bị hủy khi method hoàn thành
* Biến instance: Là các biến trong một lớp nhưng bên ngoài phương thức. các biến này được khởi tạo khi lớp được khởi tạo. Các biến sơ thẩm có thể được truy cập từ bên trong bất kỳ method, constructor hoặc các blocks của lớp cụ thể đó.
* Biến class: được khai báo bên trong một lớp, bên ngoài bất kỳ phương thức nào với từ khóa static.

# Java Regex

* Java Regex là một API để định nghĩa một mẫu để tìm kiếm hoặc thao tác với chuỗi. Nó được sử dụng rộng rãi để xác định ràng buộc trên các chuỗi như xác thực mật khẩu, email, kiểu dữ liệu datetime

# Exception

* Exception là một sự kiện xảy ra trong quá trình thực thi một chương trình Java, nó làm phá vớ cái flow bình thường của một chương trình, thậm chí chết chương trình…
* Checked exception: Là loại exception xảy ra trong lúc compile time, nó cũng có thể được gọi là **compile time exceptions**. Loại exception này không thể bỏ qua được trong quá trình compile, bắt buộc ta phải handle nó.
* Ví dụ: IOException, FileNotFoundException, NoSuchFieldException, …
* Unchecked exception
* Là loại exception xảy ra tại thời điểm thực thi chương trình, nó cũng có thể gọi là runtime exception đó là programming bugs, lỗi logic của chương trình… Loại exception này được bỏ qua trong quá trình compile, không bắt buộc ta phải handle nó.

# Data Structures

Các cấu trúc dữ liệu cung cấp bởi các package tiện ích của Java rất mạnh mẽ và thực hiện các tính năng rộng rãi. Những cấu trúc dữ liệu này bao gồm những interface và class.

* Lớp Enumeration trong java:  Enum trong Java định nghĩa các phương thức từ đó bạn có thể liệt kê (từng phần tử tại một thời điểm) các phần tử trong một tập hợp các đối tượng.
* Lớp Bitset trong java: Lớp Bitset trong java dùng để triền khai 1 nhóm các bit hoặc flag mà có thể được thiết lập và xóa một cách riêng rẽ. Class này rất hữu dụng khi bạn muốn lưu trữ 1 tập các giá trị Boolean và chỉ muốn gắn từng bit các giá trị và thiết lập hoặc xóa nó thích hợp.
* Lớp Dictionary là một abstract class để định nghĩa cấu trúc dữ liệu cho việc liên kết giữa các key tới value. Nó hữu ích trong các trường hợp truy cập dữ liệu thông qua key cụ thể thay vì sử dụng một index.
* Hashtable: Cung cấp các ý nghĩa về mặt tổ chức dữ liệu dựa vào cấu trúc mà người dùng định nghĩa key.
* Properties : là lớp con của Hashtable. Nó được sử dụng để duy trì danh sách các giá trị trong đó key là String và Value cũng là một String.

# Collection: Java Collection Framwork ( có thể gọi là nền tảng tập hợp) được xây dựng các interface định nghĩa các thao tác với tập hợp, các class cụ thể thực thi các interface và các giải thuật thông dụng thường xuyên được sử dụng với tập hợp.



## Set Interface

Set (tập hợp) là kiểu dữ liệu mà bên trong nó mỗi phần tử chỉ xuất hiện duy nhất một lần( tương tự tập hợp trong toán học) . Một số class thực thi Set Interface thường gặp:

* TreeSet: là 1 class thực thi giao diện Set Interface, trong đó các phần tử trong set đã được xắp sếp
* HashSet: là 1 class implement Set Interface, mà các phần tử được lưu trữ dưới dạng bảng băm ( hash table)

## List Interface

* Array List:

+) Là class triển khai của List interface  
+) Sử dụng cấu trúc mảng để lưu phần tử.  
+) Các phần tử trong array list có thể trùng nhau  
+) Dùng array list khi truy xuất phần tử nhiều hơn là cập nhật và xóa. ( bởi vì mỗi khi thêm hoặc xóa , AL sẽ coppy mảng đấy sang mảng mới hoàn toàn và phải cập nhật lại mảng  
+) các phần tử có thể trùng nhau.

* Linked List: Danh sách liên kết đôi. Lưu dữ liệu dạng node   
  +) Cập nhật và xóa phần tử nhanh hơn  
  +) Có thể trùng nhau
* Vector: có cách lưu trữ như ArrayList tuy nhiên khác Vector là đồ bộ, có thể hoạt động đa luồng mà không cần gọi synchronize một cách tường minh
* Stack: có cách hoạt động dựa trên cơ sở cấu trúc dữ liệu ngăn xếp(stack) với kiểu LIFO.

## Queue Interface: Nổi tiếng với kiểu dữ liệu FIFO,

* Linked List
* PriorityQueue: là 1 dạng queue mà trong đó các phần tử trong queue sẽ được sắp xệp
* ArrayDeque: là 1 dạng deque được implement dựa trên nhiều mảng.

## Map interface

* TreeMap: là clas s thực thi Map interface với dạng cây đỏ đen (Red – Black tree) trong đó các key đã được sắp xếp. Class này cho phép thời gian thêm, sửa , xóa, tìm kiếm 1 phần tử trong mảng là tương đương nhau là O(log(n))
* HashMap: các key được lưu trữ dưới dạng bảng băm, cho phép tìm kiếm nhanh 0(1).
* EnumMap: các key trong Map là các enum chứ khoogn phải object như các dạng Map trên.
* WeakHashMap: tương tự HashMap , tuy nhiên có điểm đáng chú ý là các key của Map chỉ là các Weak reference( hay Weak key), có nghĩa phần tử sẽ bị xóa khi key được giải phóng hay không còn 1 biến nào tham chiếu đến key nữa.

# Generic:

* Generic trong java là một cách để xác định các kiểu cụ thể cho các lớp và phương thức trong ngữ cảnh khác nhau.
* Quy ước đặt tên tham số: ( không bắt buộc) :  
  +) E: Element (phần tử)  
  +) K – Key   
  +) V – Value  
  +) N – Number ( kiểu số: Integer, Double, Float,…)  
  +) T – Type (Kiểu thuộc Wrapper class: String, Integer, Long, …)  
  +) S, U, V – được sử dụng để đại diện cho các kiểu Type thứ 2, 3, 4, …

# Serialization trong java: Khi lưu đối tượng xuống file hoặc truyền đối tượng qua mạng thì ta phải chuyển đối tượng thành kiểu dữ liệu dạng chuỗi thì mới được.

# Thread (đa luồng) : trong cùng một thời điểm thì có thể xử lý nhiều công việc khác nhau, chạy song song với nhau. Có 2 cách tạo thread là implements Runable() hoặc extends Thread.

Chu kỳ sống của thread: 1 thread được hình thành ở trạng thái new. Khi chúng ta start() thì nó chuyển sang Runable, xuống Running sau trạng thái Running thread có thể kết thúc, tuy nhiên trong trường hợp khác từ Running có thể chuyển sang trạng thái Non-runable bởi vị Thread có thể ngủ đông hoặc là bị khóa bởi thread khác. 