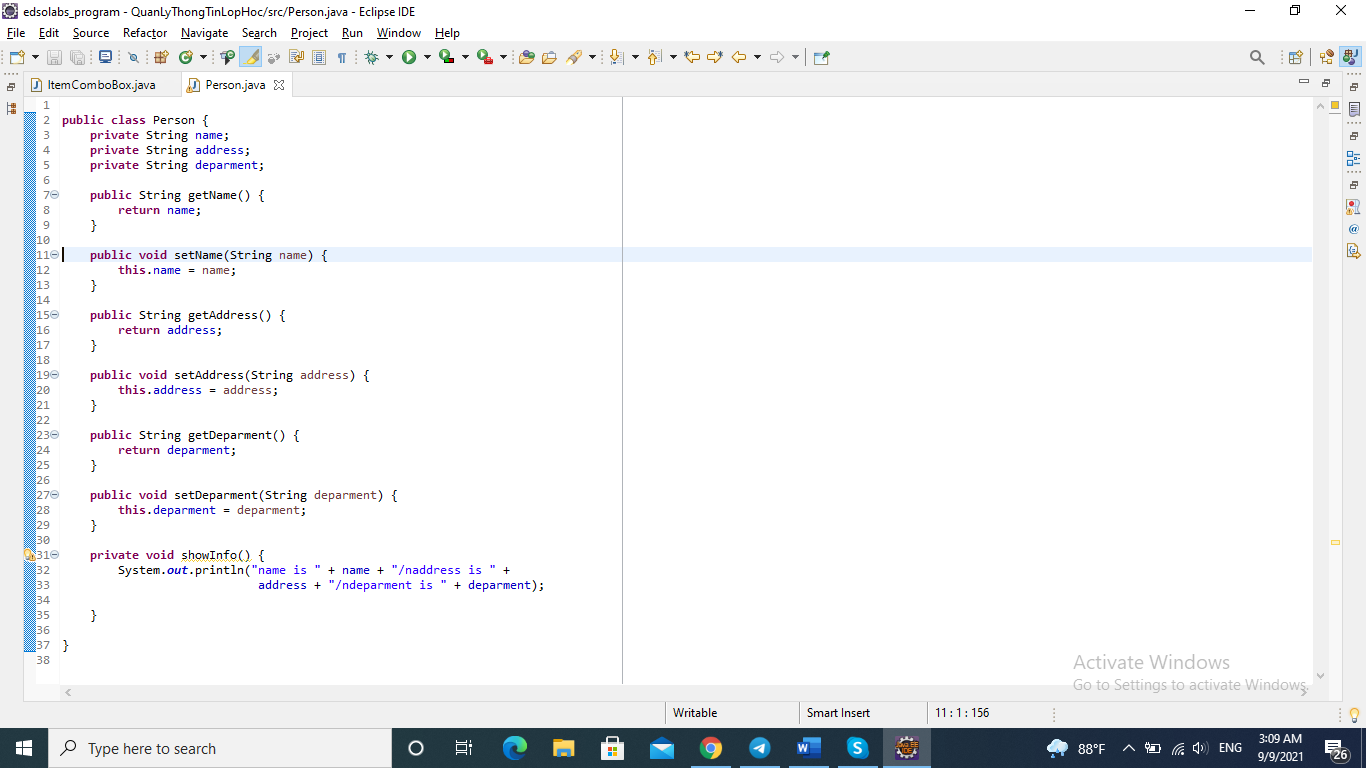
BÁO CÁO FESHER STRAINING

1. Lập trình hướng đối tượng
2. Lập trình hướng đối tượng

* Khái niệm:
* Lập trình hướng đối tượng là một phương pháp lập trình dựa trên các đối tượng được trừu tượng từ trong cuộc số.
* Đối tượng(Object) là một thực thể( cso thể là vật, người, ……)có hành vi.

VD:

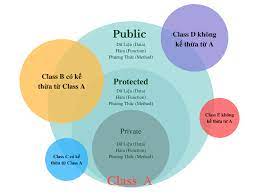


* Lớp(Class) là tập hợp các đối tượng có cùng chung một số đặc điểm.
* Đặc trưng của lập trình hướng đối tượng
* Tính đóng gói(**Encapsulation**)
* Là trạng thái của đối tượng mà các thuộc tính hay method đc bảo vệ ở các mức khác nhau, cho phép môi trường bên ngoài tác động tới các thuộc tính của đối tượng trong phạm vi cho phép (là phạm vi truy cập – access modifier).
* Tính đóng gói giúp cho đối tượng che dấu dữ liệu đảm bảo tính tàn vẹn dữ liệu bảo mật của đối tượng.
* Phạm vi truy cập trong tính đóng gói(access modifier). Tính đóng gói có các loại phạm vi truy cập như public, protected, private.

+ Public bất kỳ class nào cũng có thể truy cập từ bất kỳ đâu

+ Protected các thuộc tính có phạm vi truy cập là protected có thể truy cập được bởi chính class chứa thuộc tính này và những class con của class chứa thuộc tính này.

+ Private các thuộc tính có phạm vi truy cập là private chỉ có thể truy cập trong chính class chứa thuộc tính đó.



* VD: Class Person có các thuộc tính: name, address, deparment có mức độ truy cập là private và các method: getName(), setName(), getAddress(), setAddress(), getDeparment(), setDeparment() có phạm vi truy cập là public. Khi khởi tạo 1 đối tượng Person ta không thể gán giá trị trực tiếp cho các thuộc tính name, address, deparment, muốn gán được giá trị cho các thuộc tính trên chỉ có thể thông qua các method cso phạm vi truy cập là public.
* Tính kế thừa(**Inheritance**)
* cho phép xây dựng lớp mới dựa trên lớp cũ đã có.
* Lớp con kế thừa các thuộc tính và phương thức public và protected của lớp cha qua từ khóa extends, lớp con có thể sử dụng lại hoặc sửa đổi các phương thức của class cha và khai báo them các thuộc tính, phương thức của riêng mình.
* Kế thừa giúp tái sử dụng code, tạo ra một khung basic để xây dựng các đối tượng.
* Các kiểu kế thừa trong java:

+ **Kế thừa đơn**(Single Inheritance) là lớp con chỉ kế thừa duy nhất 1 lớp cha.

+ Ví dụ:

Class HocSinh kế thừa Class Person gọi là kế thừa đơn do class HocSinh chỉ kế thừa một class Person là class cha.

+ **Đa kế thừa**(Mutilevel Inheritance) là một lớp con có thể kế thừa nhiều class cha, loại kế thừa này có trong c++ nhưng trong java thì không có.

+ Ví dụ: Class SinhVien Vừa kế thừa Class Person vừa kế thừa Class NguoiDan.

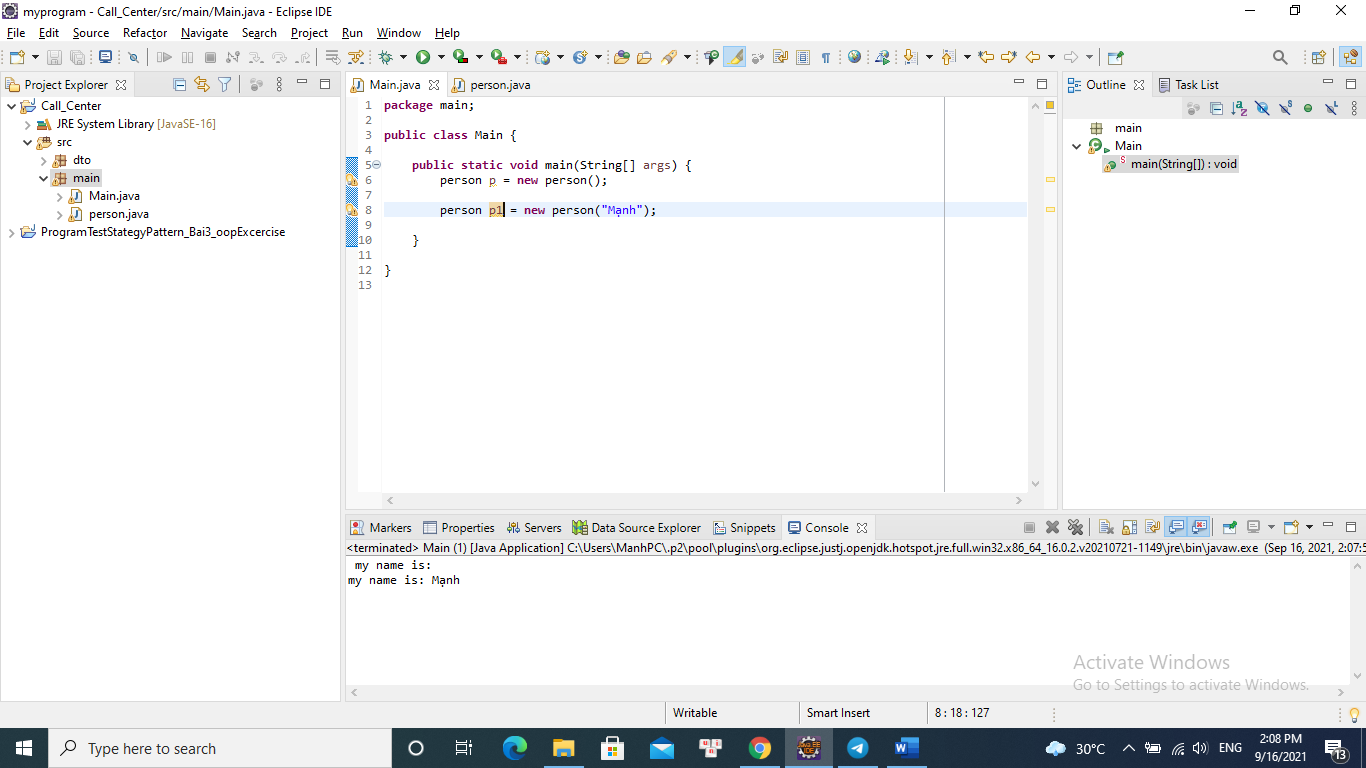
+ **Kế thừa phân cấp**(Hierarchical Inheritancce) là loại kế thừa mà nhiều lớp con kế thừa từ 1 lớp cha.

+ Ví dụ: Class HocSinh và Class GiaoVien cùng kế thừa từ một Class Person.

* Tính đa hình(**Polymorphism**)

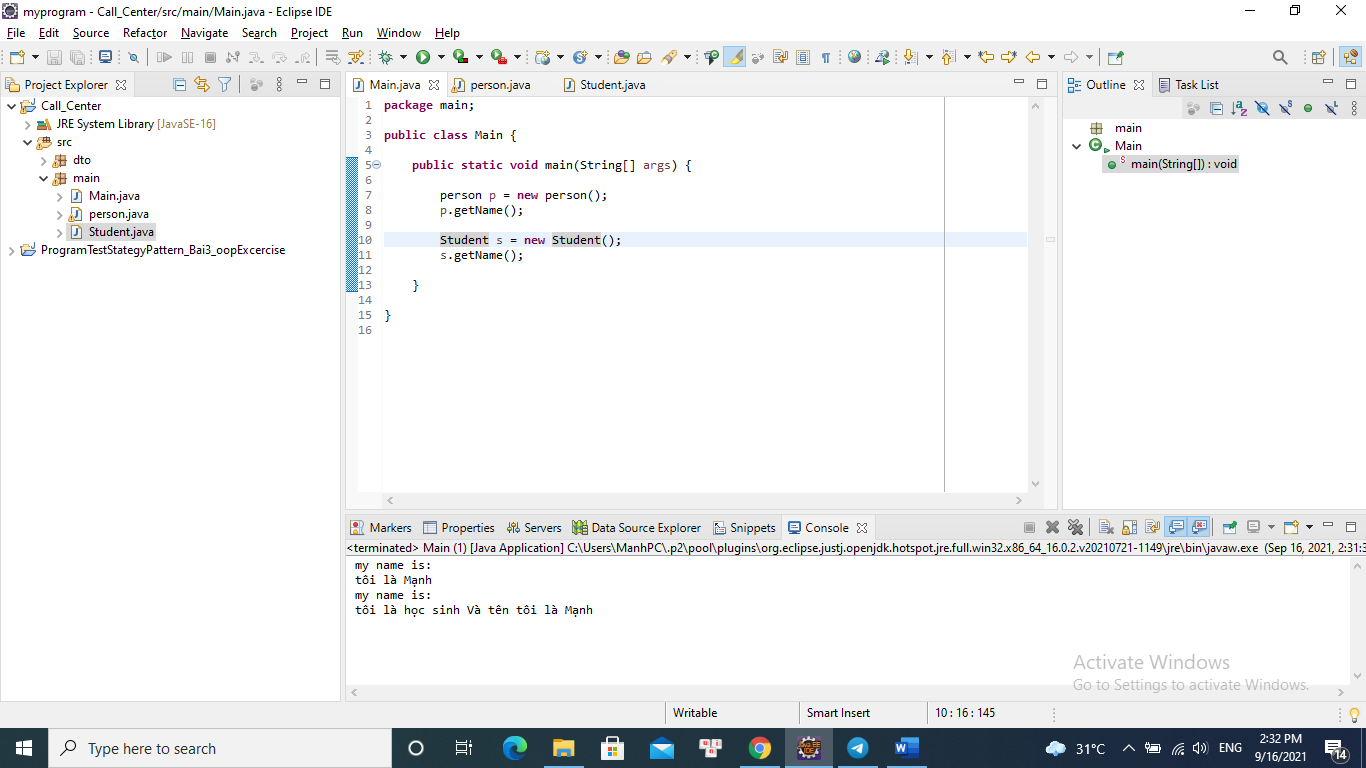
+ Là khả năng đối tượng có thể một tác vụ theo nhiều cách khác nhau. Trong Java ta sử dụng nạp trồng phương thức(method overloading) và ghi đè phương thức(method overriding) để thu được đa hình.

+ Method overloading cho phép có nhiều phương thức cùng tên nhưng khác nhau về đối số chuyền vào. Việc các phương thức cùng tên cùng tham số chuyền vào và khác nhau về kiểu trả về sẽ không thể khởi tạo vì máy ảo JVM không thể phân biệt được các method với nhau.

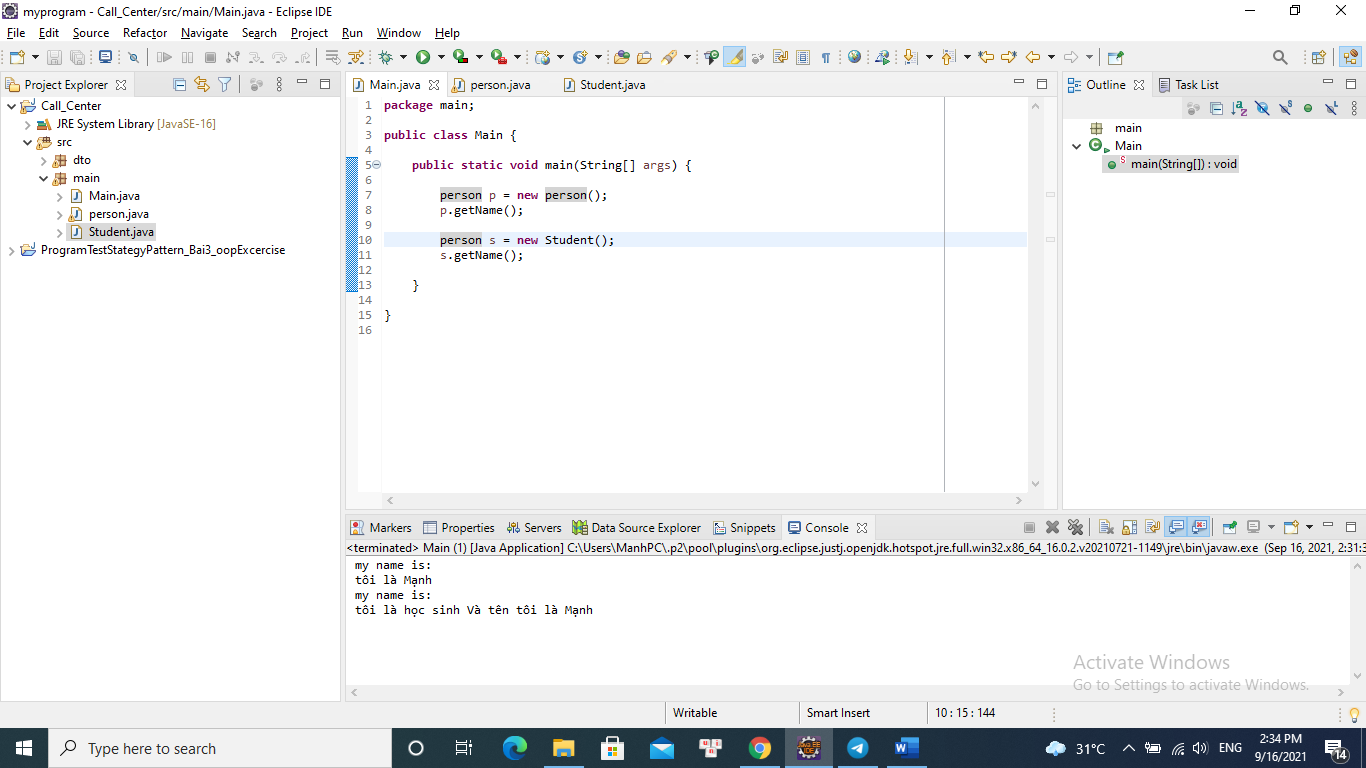
+ Ví dụ: 

+ Method overriding là lớp con khi kế thừa một lớp cha và ghi đè lại một phương thức cùng tên cùng đối số chuyền vào cùng kiểu trả về của lớp cha. Không thể ghi đè phương thức static vì phương thức static là phương thức thuộc về class.

+ Ví dụ:



+ Đa hình Runtime ví dụ:



* Tính trừu tượng(Abstraction)

+ Tính trừu tượng dung để chỉ quá trình ẩn đi việc chiển khai một số tính năng khỏi người dung. Ví dụ như bài toán giải phương trình bậc hai người dung chỉ quan tâm đến các hệ số chuyền vào và kết quả hai nghiệm chứ không quan tâm đến tính denta như thế nào hay công thức nghiệm sử dụng ra sao, vậy nên tính trừu tượng giúp lập trình viên che đi những sử lý này.

+ Thông qua qua việc che dấu đi những xử lý không cần thiết nên làm cho code đơn giản hơn, tăng tính hiệu quả của ứng dụng.

+ Một số lợi ích của việc sử dụng tính trừu tượng:

* Gao diện người dung đơn giản.
* Aản đi những đoạn code phức tạp.
* Nâng cao tính bảo mật.
* Việc bảo trì dễ dàng hơn.
* Việc cập nhật một số this năng sẽ giảm sự ảnh hưởng đến các đối tượng khác.

+Lớp trừu tượng(abstract class) là class được sinh ra để làm khung sườn cho những class khác dựa theo đó xây dựng. Cú pháp [accessmodifier] abstract class [tên lớp]() {

Tập câu lệnh

}

+ Hàm khởi tạo(ConStructor) là phương thức đặc biệt không có kiểu dữ liệu có tên trùng với tên class, một class có thể có nhiều hàm khởi tạo mỗi hàm khác khác nhau về đối số chuyền vào. Hàm khởi tạo dung để khởi tạo đối số đầu vào cho đối tượng. Cú pháp [accessmodifier] class [ten class](danh sách đối số ){

Tập lệnh

}

+ Hàm hủy(Deconstructor) là hàm dung để hủy hay xóa một đối tượng.

+ Phương thức trừu tượng(abstract method) là một phương thức có khai báo là abstract, không có thân phương thức. phương thức trừu tượng này được dung khi một abstract class có những phương thức muốn class con khi kế thừa bắt buộc phải override của class cha. Việc này đảm bảo cho class con kế thừa theo khung sườn đã thiết kế của class cha.

+ Giao diện(Interface) là một class đặc biệt được dung để thu được đa hình và đa kế thừa trong java(vì java không cho phép một class con kết thừa nhiều class cha nên interface sinh ra để giải quyết vấn đề này). Các phương thức được khai báo trong interface đều là phương thức trừu tượng. Interface không phải là một class đúng nghĩa interface không thể tạo đối tượng.

1. Tìm hiểu hai tiêu chí đánh giá chất lượng thiết kế module: module conhension & coupling

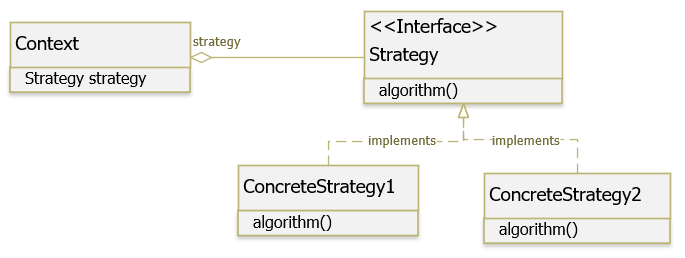
* Conhesion và coupling là hai thuộc tính luôn đi cùng nhau và hight conhesion và low coupling là đích đến mà lập trình viên hướng tới.
* Conhension: là thuộc tính đề cập đến nhiệm vụ của từng module. Nhiệm vụ của từng module càng rõ rang càng tách biệ thì cohesion càng cao(hight cohesion).
* Coudling: là thuộc tính đề cập đến vấn đề phụ thuộc lẫn nhau giữa các component. Nó có nghĩa là các component ít phụ thuộc vào nhau sự thay đổi của component này it hoặc là không thay đổi hay ảnh hưởng đến các component khác. Và khi lập trình ta cần hướng đến mục đích giảm thiểu tối đa sự ảnh hưởng của các component đến nhau(low coudling).

1. Xử lí ngoại lệ

* Ngoại lệ là một lỗi xảy ra khi chương trình đang đươc chạy, nó phá vỡ cấu trúc bình thường của chương trình là cho chương tình không thể tiếp tục vận hành.
* Nguyên nhân xảy ra ngoại lệ có thể do nhiều lí do khác nhau nằm ngoài dự tính của chương trình. Các lỗi này xay ra có thể do lập trình viên không lường trước được tất cả cá trường hợp xảy ra, cũng có thể đến từ phía người dùng và lỗi do dữ liệu vật lý.
* Ví dụ:
* Người dung cố tình nhập dữ liệu sai(sai kiểu dữ liệu, số quá lớn vượt qua giới hạn kiểu dữ liệu).
* Trong tính diện tích tam giác lập trình viên bỏ qua trường hợp ba cạnh không hợp thành một tam giác.
* Bộ nhớ bị đầy.

1. Design pastern

* State Pattern
* Stategy Pattern
* Khái niệm stategy pattern là một trong những pattern thuộc nhóm hành vi(Pattern Behavior). Nó cho phép định nghĩa tập hợp các thuật toán, đóng gói từng thuật toán và dễ dàng thay đổi các thuật toán một cách đọc lập không ảnh hưởng đến đối tượng.
* Tác dụng các stategy giúp cho tách rời được phần chức năng xử lý và đối tượng ra riêng biệt. Nó được dung như để thay thế tính kế thưa.
* Mô hình cài đặt của stategy pattern



Trong đó Class Context chứa tham chiếu đến đối tượng Stategy và nhận các yêu cầu từ phiwas client và chuyển cho Stategy thực hiện.

Stategy định nghĩa một hành động nào đó của đối tượng Context.

ConcreteStategy1,2 là các hành động cự thể của Stategy.

1. Java core
2. Tìm hiểu các mô hình Threads Pool(Pooling Threads model)