



MagRabbit®

Java Core DAO, DTO

Author: Giau Le

- <https://viettuts.vn/java>
- <https://www.w3schools.com/java/>
- <https://o7planning.org/vi/12571/lich-su-cua-java-va-su-khac-biet-giua-oracle-jdk-va-openjdk>
- Github
 - <https://github.com/giaule91/java-core-demo.git>

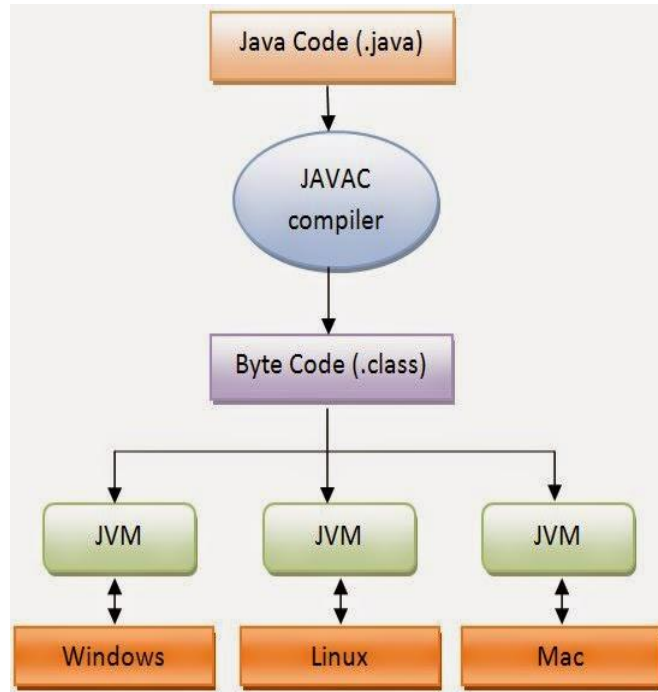
- 1. Overview and setup**
- 2. Rule & syntax**
- 3. Object and Classes**
- 4. Data Types**
- 5. Operator, Decision Making, Loop**
- 6. OOP**
- 7. Collections**
- 8. Exception**
- 9. DAO, DTO**



Overview and setup

- Java programming language was originally developed by Sun Microsystems
- Java built to suit various types of platforms.
 - For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.
- Java is guaranteed to be **Write Once, Run Anywhere.**

Overview and setup





Overview and setup

- Jdk 8
 - <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
 - Install
 - Set JAVA_HOME:
 - Select Control Panel and then System.
 - Click Advanced and then Environment Variables.
 - Under System Variables, click New.
 - Enter the variable name as JAVA_HOME.
 - Enter the variable value as the installation path of the JDK
 - Ex: `JAVA_HOME = C:\Program Files\Java\jdk1.8.0_251`
 - Click OK.
 - Click Apply Changes.

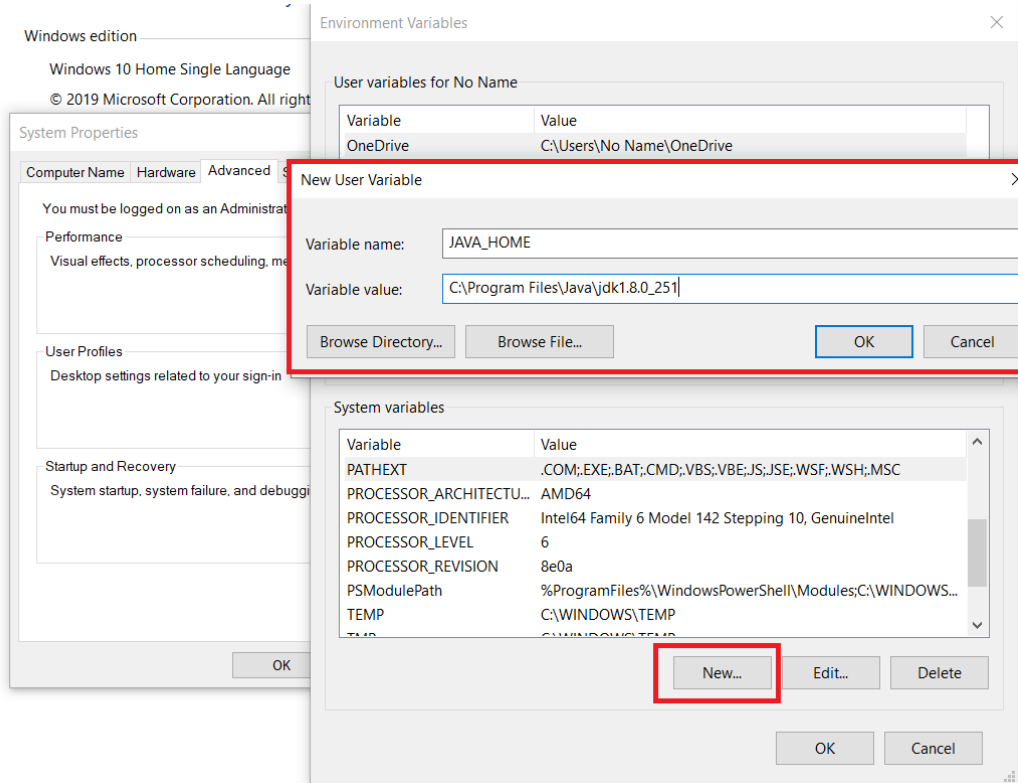


Overview and setup

- Update PATH:
 - In System variables, find PATH
 - Add: `%JAVA_HOME%\bin;`
- To see if it worked, open up the Command Prompt and type `java -version` and see if it prints your newly installed JDK.
 - `java -version`

Overview and setup

- Device Manager
- Remote settings
- System protection
- Advanced system settings



The screenshot shows the Windows System Properties dialog box with the 'Advanced' tab selected. The 'Environment Variables' button is highlighted, and the 'Environment Variables' dialog box is open. In the 'Environment Variables' dialog, the 'User variables for No Name' tab is active, showing a table with one variable: 'OneDrive' with value 'C:\Users\No Name\OneDrive'. The 'New User Variable' dialog box is also open, showing 'Variable name: JAVA_HOME' and 'Variable value: C:\Program Files\Java\jdk1.8.0_251\'. The 'System variables' tab is also visible, showing a table with various system variables. The 'New...' button in the 'System variables' dialog is highlighted.

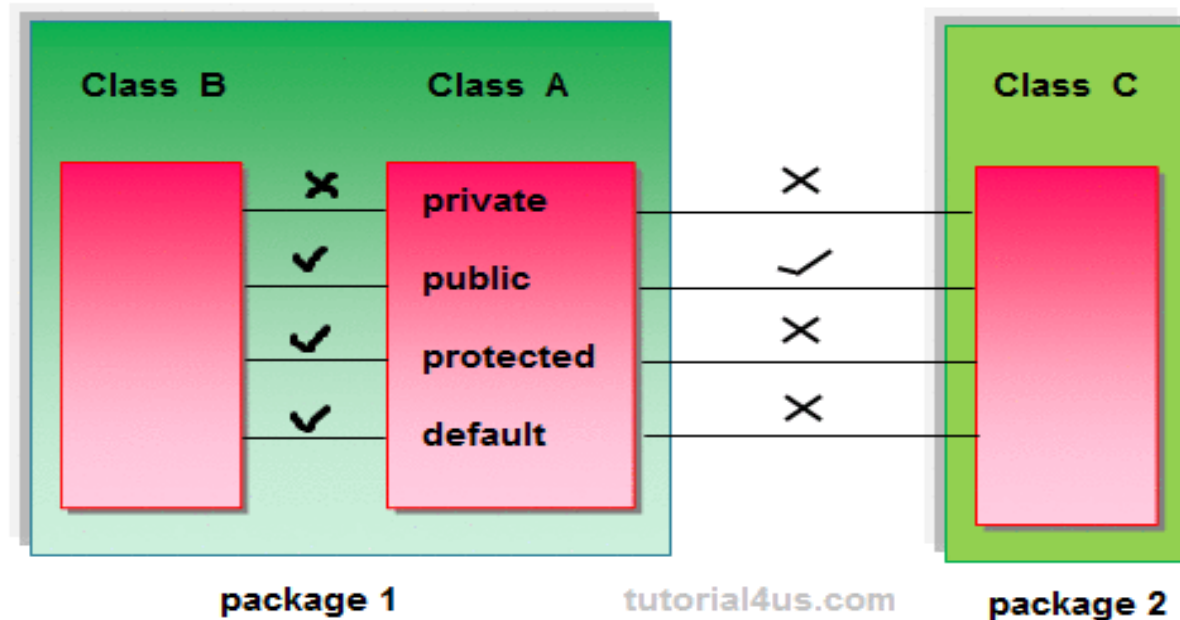
Variable	Value
OneDrive	C:\Users\No Name\OneDrive

Variable name:	Value
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_251\

Variable	Value
PATH	COM;EXE;BAT;CMD;VBS;VBE;JS;JSE;WSF;WSH;MSC
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 142 Stepping 10, GenuineIntel
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e0a
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\WINDOWS...
TEMP	C:\WINDOWS\TEMP
TMP	C:\WINDOWS\TEMP

- **Case Sensitivity:** **Hello** and **hello** would have different meaning in Java
- **Class Names:** For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.
 - **Example:** *class MyFirstJavaClass*
- **Method Names:** All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.
 - **Example:** *public void myMethodName()*

- **Access Modifiers** – default, public, protected, private



	Class	Package	Subclass (same pkg)	Subclass (diff pkg)	World
public	+	+	+	+	+
protected	+	+	+	+	
no modifier	+	+	+		
private	+				

+ : accessible

blank : not accessible



- Local Variables
- Class Variables (Static Variables)
- Instance Variables (Non-static Variables)

Local and Instance Variables

```
public class Car
{
    private double gasInTank;
    ...
    public void drive(double distance)
    {
        double gasConsumed = distance / milesPerGallon;
        gasInTank = gasInTank - gasConsumed;
    }
    ...
}
```

Handwritten annotations:

- instance variable* (with an arrow pointing to `gasInTank`)
- local variable* (with an arrow pointing to `gasConsumed`)



Static Variables - Example

■ Using *static* variables:

```
public class Circle {
    // class variable, one for the Circle class, how many circles
    private static int numCircles = 0;
    private double x,y,r;

    // Constructors...
    Circle (double x, double y, double r){
        this.x = x;
        this.y = y;
        this.r = r;
        numCircles++;
    }
}
```

- Từ khóa final
 - **Biến final:** bạn không thể thay đổi giá trị của biến final (nó sẽ là hằng số).
 - **Phương thức final:** bạn không thể ghi đè phương thức final.
 - **Lớp final:** bạn không thể kế thừa lớp final.

```
1 public class FinalExam1 {
2     final int MAX_SPEED = 90; // biến final
3
4     void run() {
5         MAX_SPEED = 400;
6     }
7
8     public static void main(String args[]) {
9         FinalExam1 obj = new FinalExam1();
10        obj.run();
11    }
12 }
```

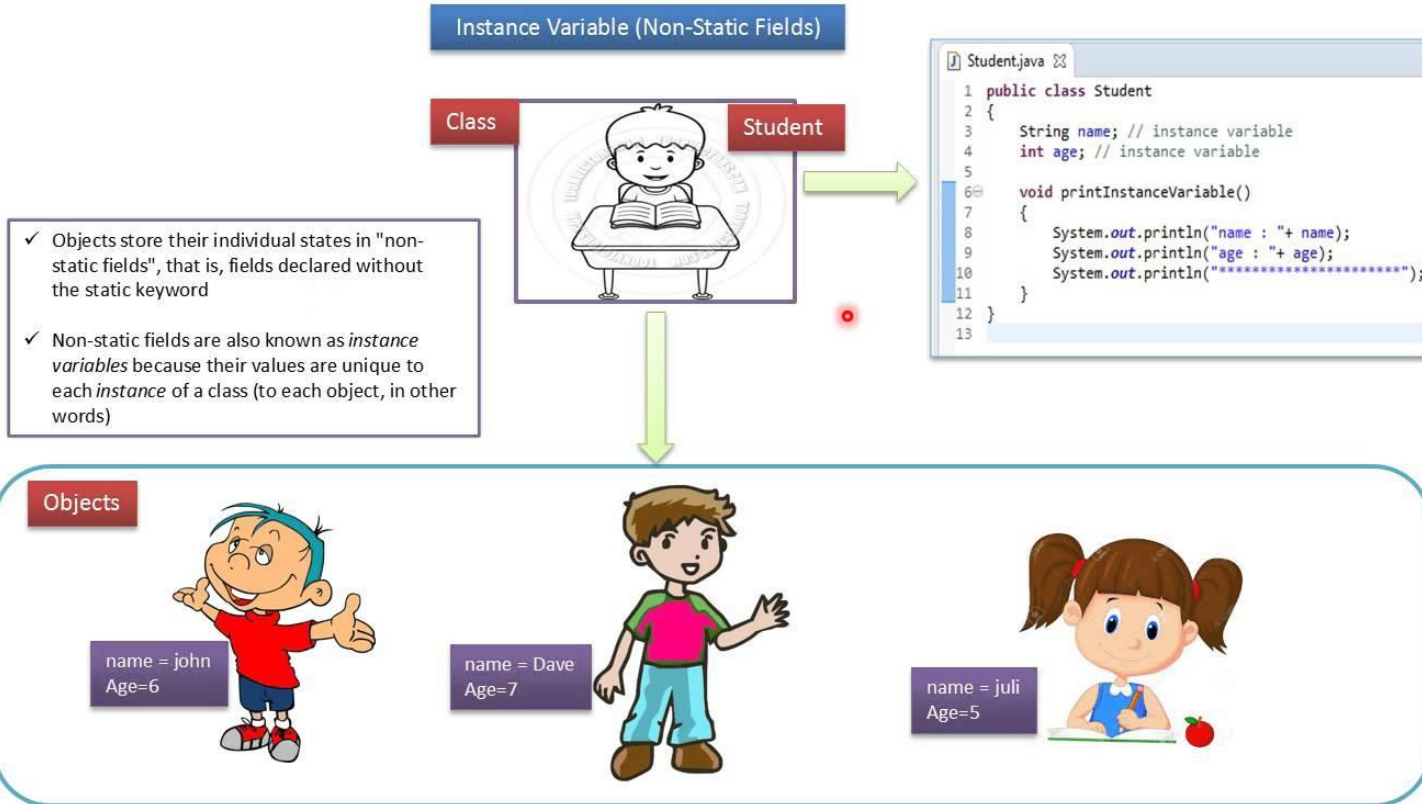
```
1 class Bike {
2     final void run() {
3         System.out.println("running");
4     }
5 }
6
7 public class SH extends Bike {
8     void run() {
9         System.out.println("Chay an toan voi 150km/h");
10    }
11
12    public static void main(String args[]) {
13        SH sh = new SH();
14        sh.run();
15    }
16 }
```

```
final class Bike {
}

public class SH1 extends Bike {
    void run() {
        System.out.println("Chay an toan voi 150km/h");
    }

    public static void main(String args[]) {
        SH1 sh = new SH1();
        sh.run();
    }
}
```

Object and Classes





Object and Classes

- **Object** – Objects have states and behaviors.
 - Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class.
- **Class** – A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support.

- Constructors
 - Each time a new object is created, at least one constructor will be invoked.
 - The main rule of constructors is that they should have the same name as the class.
 - A class can have more than one constructor.
 - Every class has a least of constructor.
 - If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.

```
public class SubClassOne{  
    //Constructor block  
    SubClassOne(){  
        System.out.println("Learn from examples");  
    }  
    public static void main(String[] args) {  
        SubClassOne classOne = new SubClassOne();  
    }  
}
```


- There are two data types available in Java
 - Primitive Data Types
 - Reference/Object Data Types
- https://www.w3schools.com/java/java_data_types.asp

- Java cung cấp các loại toán tử sau để thao tác với biến
 - Arithmetic Operators
 - Relational Operators
 - Bitwise Operators
 - Logical Operators
 - Assignment Operators
 - Misc Operators
- https://www.w3schools.com/java/java_operators.asp

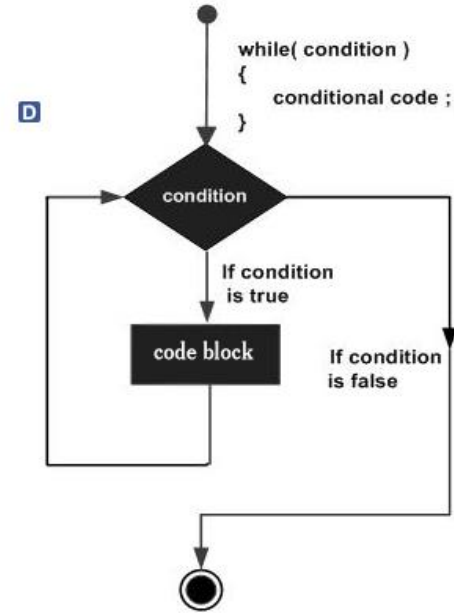
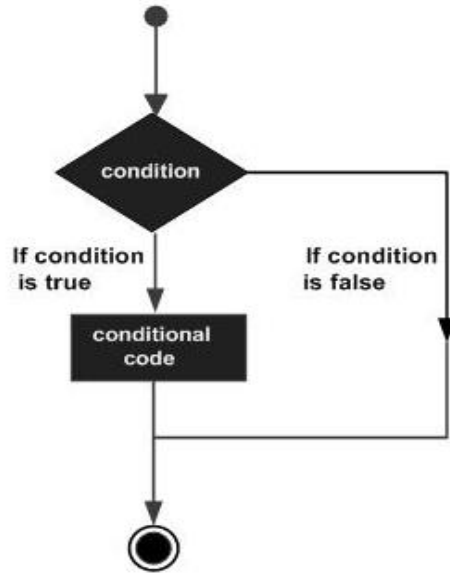


Operator, Decision Making, Loop

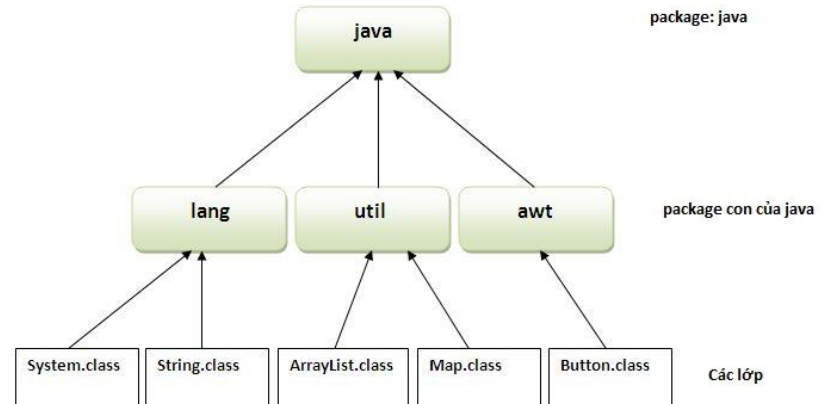
- Java provides a rich set of operators to manipulate variables
 - Arithmetic Operators
 - Relational Operators
 - Bitwise Operators
 - Logical Operators
 - Assignment Operators
 - Misc Operators
- https://www.w3schools.com/java/java_operators.asp



Operator, Decision Making, Loop



- **Package trong java** có thể được phân loại theo hai hình thức, package được dựng sẵn và package do người dùng định nghĩa.
 - Có rất nhiều package được dựng sẵn như java, lang, AWT, javax, swing, net, io, util, sql, ...

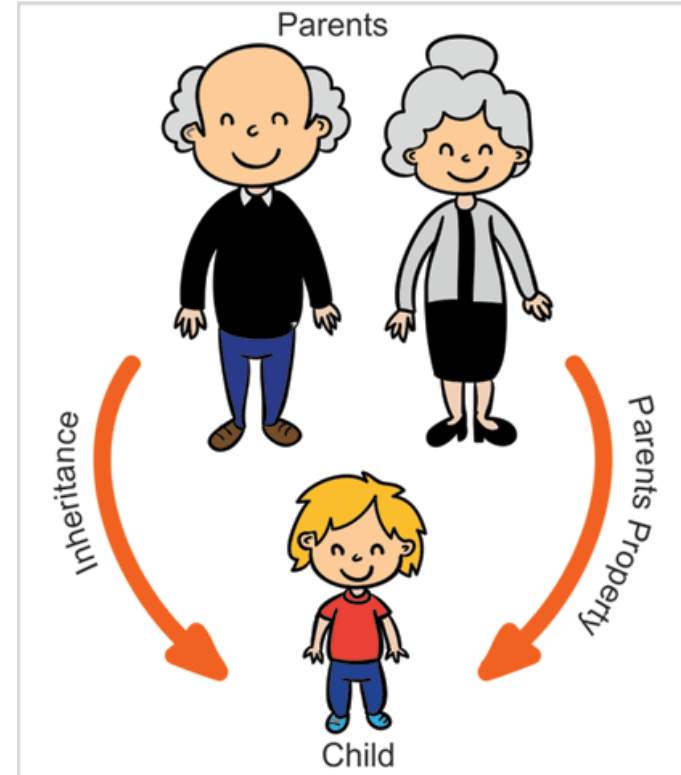
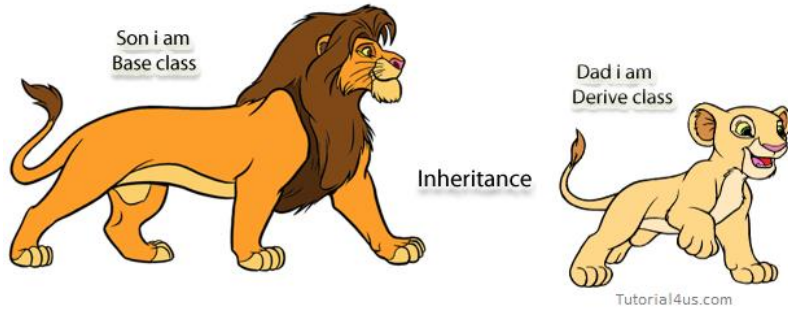


- Tạo package **com.demo.core.lab1**
- Tạo class mô tả dưới đây:
 - Class Teacher với các thuộc tính **name**, Class Subject với các thuộc tính name, classId
 - Có thể tạo 1 teacher với name và age, subject (dùng contrustor)
 - Có thể tạo 1 teacher với subject (dùng contrustor)
 - Có thể tạo Subject với name
 - In ra console với:
 - Teacher Tam teaching Mathematics for Class 1
- https://www.w3schools.com/java/java_constructors.asp

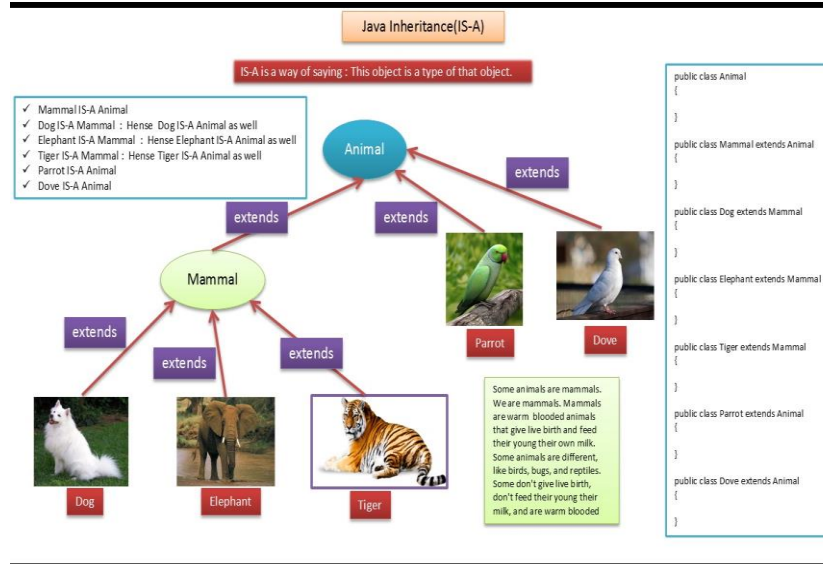
- Tạo package **com.demo.core.lab2**
- Tạo class employee(name, age, job is string, salary, department)
- Xây dựng chương trình nhập employee cho công ty, sau đó hiển thị thông tin ra console.
 - If employee job = 'developer' thì department = 'development'
 - If employee job = 'tester' thì department = 'QA'
 - Còn lại -> department = 'master'
- Note:
 - https://www.w3schools.com/java/java_user_input.asp
 - https://www.w3schools.com/java/java_conditions.asp

- Inheritance
- Overriding
- Polymorphism
- Abstraction
- Encapsulation
- Interfaces
- Packages

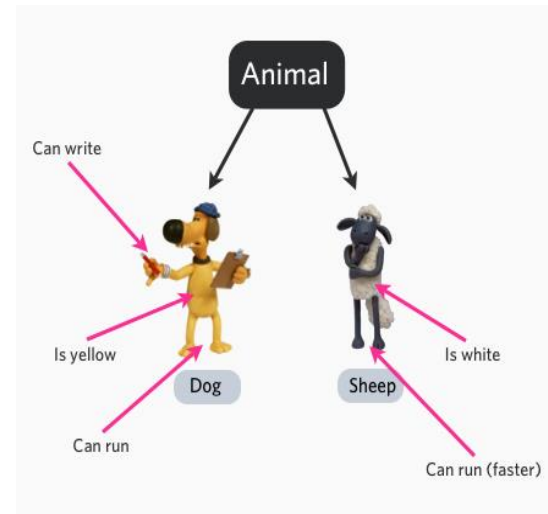
Inheritance



- How to write Inheritance on java?
 - **extends** is the keyword used to inherit the properties of a class.



- What is Overriding?
 - In object-oriented terms, overriding means to override the functionality of an existing method.



```
class Animal {
    public void move() {
        System.out.println("Animals can move");
    }
}

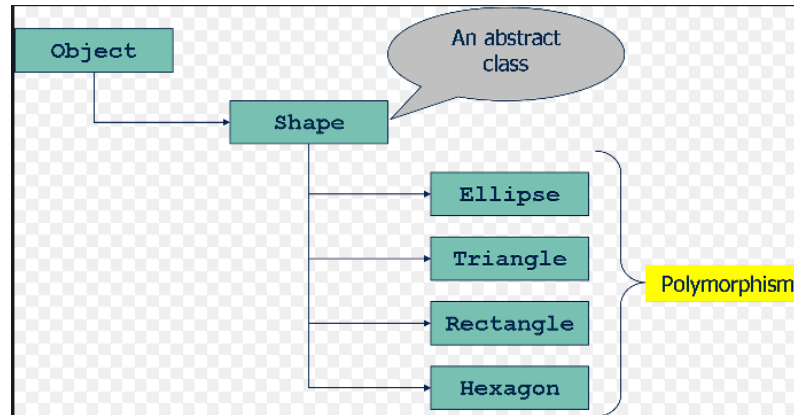
class Dog extends Animal {
    public void move() {
        super.move(); // invokes the super class method
        System.out.println("Dogs can walk and run");
    }
}

public class TestDog {

    public static void main(String args[]) {
        Dog b = new Dog(); // Dog object
        b.move(); // runs the method in Dog class
    }
}
```

Polymorphism

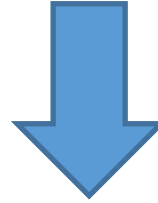
- Polymorphism is the ability of an object to take on many forms.
- The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.



Polymorphism

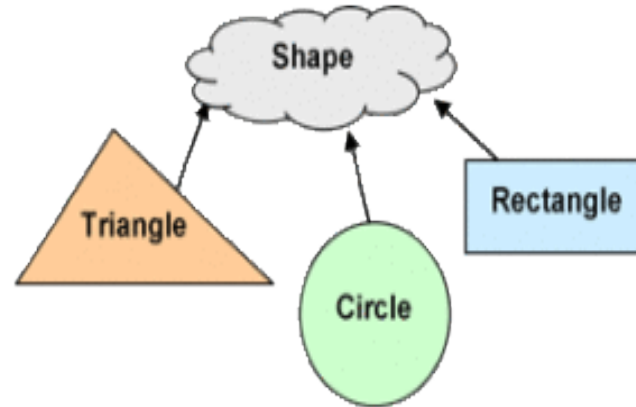
- For Requirement Example
 - A Deer IS-A Animal
 - A Deer IS-A Vegetarian
 - A Deer IS-A Deer
 - A Deer IS-A Object

```
public interface Vegetarian{}  
public class Animal{}  
public class Deer extends Animal implements Vegetarian{}
```



```
Deer d = new Deer();  
Animal a = d;  
Vegetarian v = d;  
Object o = d;
```

- Abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user.
- The user will have the information on what the object does instead of how it does it.



- Using **abstract** keyword
- Abstract classes may or may not contain *abstract methods*, i.e., methods without body (`public void get();`)
- if a class has at least one abstract method, then the class **must** be declared **abstract**.
- To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it.
- If a class is declared abstract, it cannot be instantiated.

Tạo package **com.demo.core.lab3**

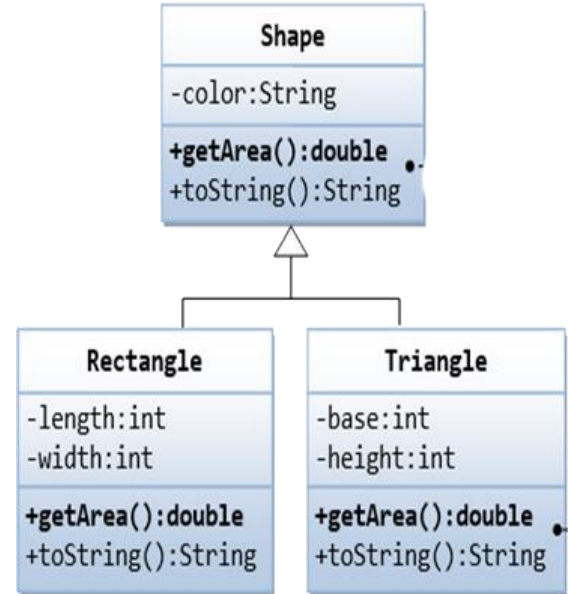
Tạo class Shape với thuộc tính color

- Method `getArea()` is abstract method

Tạo class Rectangle (hình chữ nhật) và Triangle (tam giác) là hai lớp con của Shape, với thêm 2 thuộc tính length, width

Overwrite method `getArea()` trong class Rectangle và Triangle tính diện tích tương ứng.

https://www.w3schools.com/java/java_abstract.asp





Encapsulation

- **Encapsulation** is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.
- Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit
- The variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as **data hiding**

- Benefits of Encapsulation
 - The fields of a class can be made read-only or write-only.
 - A class can have total control over what is stored in its fields.

```
/* File name : EncapTest.java */
public class EncapTest {
    private String name;
    private String idNum;
    private int age;

    public int getAge() {
        return age;
    }

    public String getName() {
        return name;
    }

    public String getIdNum() {
        return idNum;
    }

    public void setAge( int newAge) {
        age = newAge;
    }

    public void setName(String newName) {
        name = newName;
    }

    public void setIdNum( String newId) {
        idNum = newId;
    }
}
```

- Tạo get/set cho các Class ở Lab 2,3
- Set các giá trị cho các objects đó
- https://www.w3schools.com/java/java_encapsulation.asp

- An interface is a reference type in Java. It is similar to class.
- It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.
- Writing an interface is similar to writing a class. But a class describes the attributes and behaviors of an object.
- And an interface contains behaviors that a class implements.
- Unless the class that implements the interface, all the methods of the interface need to be defined in the class.

- What is different with a class
 - You cannot instantiate an interface.
 - An interface does not contain any constructors.
 - All of the methods in an interface are abstract.
 - An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.
 - An interface is not extended by a class; it is implemented by a class.
 - An interface can extend multiple interfaces.

- How to write Interfaces on Java?
 - **interface** keyword is used to declare an interface.
 - An interface is implicitly abstract.
 - Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.
 - Methods in an interface are implicitly public.

```
8 // **
9  * Interface for Product DAO.
10 * @author giaule
11 *
12 */
13 public interface ProductDAO {
14     List<ProductDTO> getAllProducts() throws SQLException;
15     boolean addNewProduct(ProductDTO newProduct) throws SQLException;
16     boolean updateProduct(ProductDTO product) throws SQLException;
17     boolean deleteProduct(ProductDTO product) throws SQLException;
18     ProductDTO getProductById(int id) throws SQLException;
19 }
20
```

- To access the interface methods, the interface must be "implemented" (kinda like inherited) by another class with the implements keyword (instead of extends)

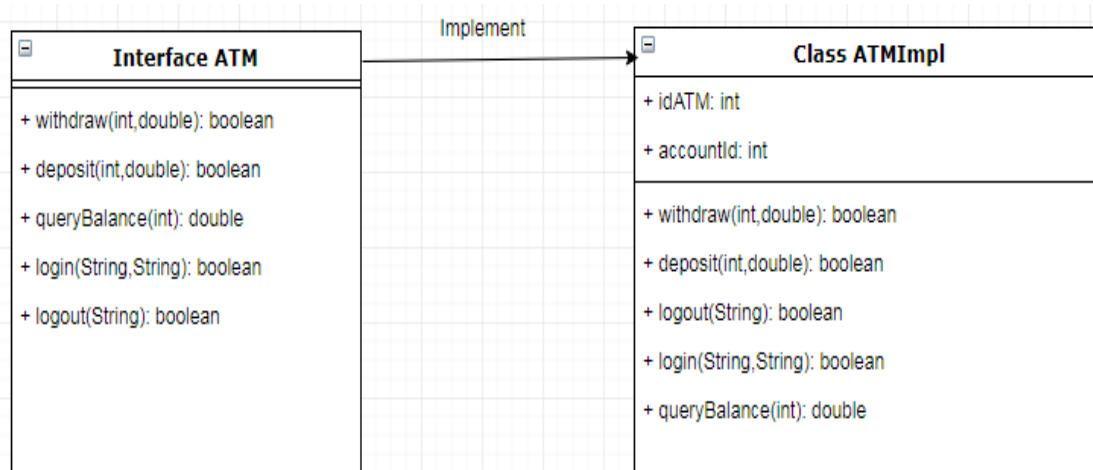
```
19 public class ProductDAOImpl implements ProductDAO{
20
21     private String jdbcURL;
22     private String jdbcUsername;
23     private String jdbcPassword;
24     private Connection jdbcConnection;
25
26
27
28+    public ProductDAOImpl(String jdbcURL, String jdbcUsername, String jdbcPassword) {}
34
35+    protected void connect() throws SQLException {}
46
47+    protected void disconnect() throws SQLException {}
52
53-    @Override
54    public List<ProductDTO> getAllProducts() throws SQLException {
55        List<ProductDTO> listProd = new ArrayList<ProductDTO>();
56
57        String sql = "SELECT * FROM product";
58
59        connect();
60
61        Statement statement = jdbcConnection.createStatement();
62        ResultSet resultSet = statement.executeQuery(sql);
```


Tạo package **com.demo.core.lab5**

Tạo 1 Interface có tên ATM

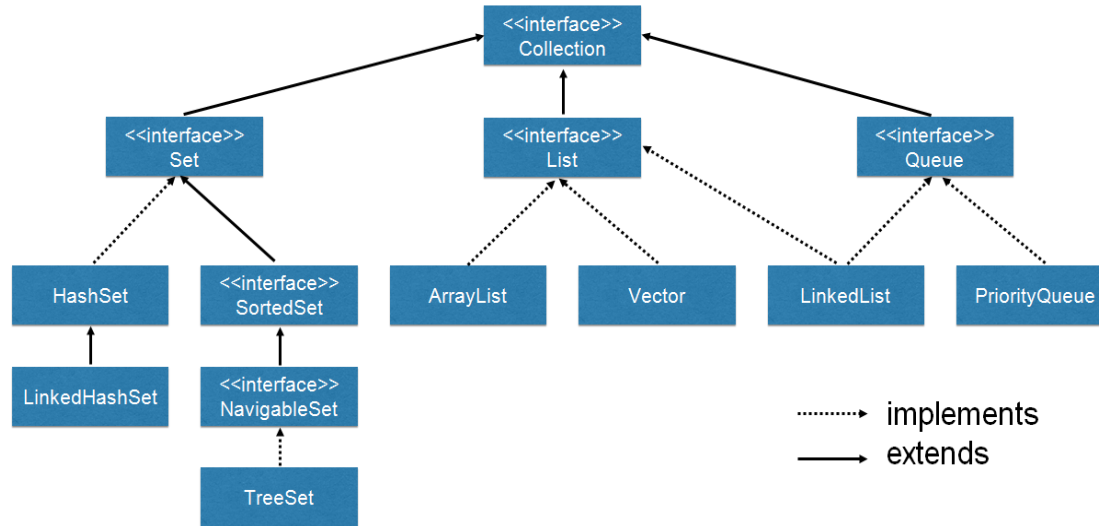
Và class implement nó có tên ATMImpl.

https://www.w3schools.com/java/java_interface.asp



- Collections is set of standard interfaces.
- The framework had to be high-performance.
- Allow different types of collections to work in a similar manner

Collection Interface



- List
 - Elements can be inserted or accessed by their position in the list, using a zero-based index.
 - A list may contain duplicate elements.

```
1 import java.util.*;
2 public class CollectionsDemo {
3
4     public static void main(String[] args) {
5         List a1 = new ArrayList();
6         a1.add("Zara");
7         a1.add("Mahnaz");
8         a1.add("Ayan");
9         System.out.println(" ArrayList Elements");
10        System.out.print("\t" + a1);
11
12        List l1 = new LinkedList();
13        l1.add("Zara");
14        l1.add("Mahnaz");
15        l1.add("Ayan");
16        System.out.println();
17        System.out.println(" LinkedList Elements");
18        System.out.print("\t" + l1);
19    }
20 }
```



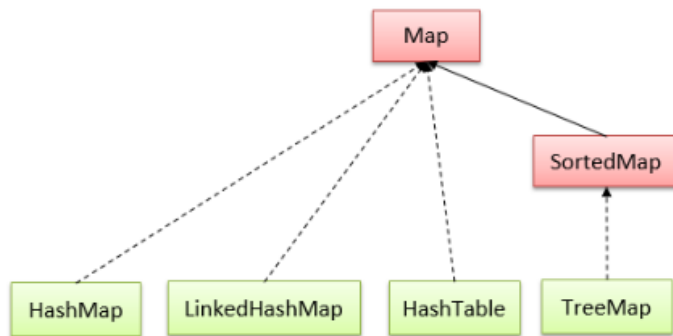
- A Set is a Collection that cannot contain duplicate elements. It models the mathematical set abstraction.

```
1 import java.util.*;
2 public class SetDemo {
3
4     public static void main(String args[]) {
5         int count[] = {34, 22,10,60,30,22};
6         Set<Integer> set = new HashSet<Integer>();
7         try {
8             for(int i = 0; i < 5; i++) {
9                 set.add(count[i]);
10            }
11            System.out.println(set);
12
13            TreeSet sortedSet = new TreeSet<Integer>(set);
14            System.out.println("The sorted list is:");
15            System.out.println(sortedSet);
16
17            System.out.println("The First element of the set is: " + (Integer)sortedSet
18                               .first());
19            System.out.println("The last element of the set is: " + (Integer)sortedSet
20                               .last());
21        }
22        catch(Exception e) {}
23    }
24 }
```

- Map
 - Maps are not *collections* in the proper use of the term, but they are fully integrated with collections.
 - Maps store key/value pairs.

```
1 import java.util.*;
2 public class CollectionsDemo {
3
4     public static void main(String[] args) {
5         Map m1 = new HashMap();
6         m1.put("Zara", "8");
7         m1.put("Mahnaz", "31");
8         m1.put("Ayan", "12");
9         m1.put("Daisy", "14");
10
11         System.out.println();
12         System.out.println(" Map Elements");
13         System.out.print("\t" + m1);
14     }
15 }
```

- Map
 - **Map** là interface thiết kế để lưu trữ cấu trúc dữ liệu theo dạng (key, value). Cả key và value đều **là** object (không chấp nhận kiểu dữ liệu primitives).



- HashMap được sử dụng để lưu trữ các phần tử dưới dạng **"key/value"**.
 - HashMap lưu trữ dữ liệu dưới dạng cặp key và value.
 - Nó chứa các key duy nhất.
 - Nó có thể có 1 key là null và nhiều giá trị null.
 - Nó duy trì các phần tử KHÔNG theo thứ tự.

```
1 import java.util.*;
2 public class CollectionsDemo {
3
4     public static void main(String[] args) {
5         Map m1 = new HashMap();
6         m1.put("Zara", "8");
7         m1.put("Mahnaz", "31");
8         m1.put("Ayan", "12");
9         m1.put("Daisy", "14");
10
11         System.out.println();
12         System.out.println(" Map Elements");
13         System.out.print("\t" + m1);
14     }
15 }
```


- Tạo package **com.demo.core.lab6**
- Quản lý khách hàng xếp hàng mua vé tại nhà ga. Thông tin lưu trữ cho khách hàng gồm: số CMND khách hàng (String), Tên khách hàng, Ga đến, giá tiền (double).
- Quản lý các mục:
 - Thêm một khách hàng mới vào hàng đợi mua vé.
 - Hiển thị các khách hàng đang xếp hàng mua vé.
 - Tìm một khách hàng theo CMND (thực hiện theo hai cách dùng list và map)
 - Khi một khách hàng đã mua vé, thì loại khách hàng này ra khỏi
- https://www.w3schools.com/java/java_user_input.asp
- https://www.w3schools.com/java/java_arraylist.asp

- An exception (or exceptional event) is a problem that arises during the execution of a program.
- When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled

- User case
 - A user has entered an invalid data.
 - A file that needs to be opened cannot be found.
 - A network connection has been lost in the middle of communications or the JVM has run out of memory.

- What kind of Exception?
 - **Checked exceptions** – A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions
 - **Unchecked exceptions** – An unchecked exception is an exception that occurs at the time of execution. These are also called as **Runtime Exceptions**.

Un-Checked Exception

```
1 public class Unchecked_Demo {  
2  
3     public static void main(String args[]) {  
4         int num[] = {1, 2, 3, 4};  
5         System.out.println(num[5]);  
6     }  
7 }
```

Checked Exception

```
1 import java.io.File;  
2 import java.io.FileReader;  
3  
4 public class FileNotFound_Demo {  
5  
6     public static void main(String args[]) {  
7         File file = new File("E://file.txt");  
8         FileReader fr = new FileReader(file);  
9     }  
10 }
```

- Catching Exceptions
 - A method catches an exception using a combination of the **try** and **catch** keywords
 - A try/catch block is placed around the code that might generate an exception.
 - Code within a try/catch block is referred to as protected code
 - Can using Multiple Catch Blocks

```
1 // File Name : Exceptest.java
2 import java.io.*;
3
4 public class Exceptest {
5
6     public static void main(String args[]) {
7         try {
8             int a[] = new int[2];
9             System.out.println("Access element three : " + a[3]);
10        } catch (ArrayIndexOutOfBoundsException e) {
11            System.out.println("Exception thrown : " + e);
12        }
13        System.out.println("Out of the block");
14    }
15 }
```

- Throws/Throw exception
 - If a method does not handle a checked exception, the method must declare it using the **throws** keyword. The throws keyword appears at the end of a method's signature.
 - You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the **throw** keyword.

```
import java.io.*;
public class className {

    public void deposit(double amount) throws RemoteException {
        // Method implementation
        throw new RemoteException();
    }
    // Remainder of class definition
}
```

- The Finally Block
 - The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.
 - Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

```
1 public class ExcepTest {  
2  
3     public static void main(String args[]) {  
4         int a[] = new int[2];  
5         try {  
6             System.out.println("Access element three : " + a[3]);  
7         } catch (ArrayIndexOutOfBoundsException e) {  
8             System.out.println("Exception thrown : " + e);  
9         } finally {  
10            a[0] = 6;  
11            System.out.println("First element value: " + a[0]);  
12            System.out.println("The finally statement is executed");  
13        }  
14    }  
15 }
```


- User-defined Exceptions
 - You can create your own exceptions in Java. Keep the following points in mind when writing your own exception classes
 - All exceptions must be a child of Throwable.
 - If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class.
 - If you want to write a runtime exception, you need to extend the RuntimeException class.

- <https://topdev.vn/blog/lap-trinh-da-luong-trong-java-java-multi-threading/>
- https://www.w3schools.com/java/java_threads.asp

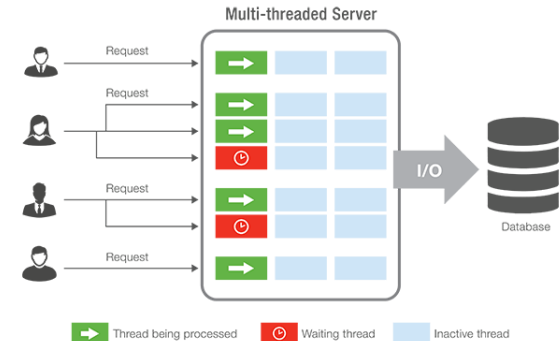
- Thread (luồng) về cơ bản là một tiến trình con (sub-process). Một đơn vị xử lý nhỏ nhất của máy tính có thể thực hiện một công việc riêng biệt. Trong Java, các luồng được quản lý bởi máy ảo Java (JVM)
- Multithreading
 - Multithreading in Java is a process of executing multiple threads simultaneously.
 - Multiprocessing and multithreading, both are used to achieve multitasking.
 - Một ứng dụng Java ngoài luồng chính có thể có các luồng khác thực thi đồng thời làm ứng dụng chạy nhanh và hiệu quả hơn.

- Advantages of Java Multithreading

- It doesn't block the user because threads are independent and you can perform multiple operations at the same time.
- You can perform many operations together, so it saves time.
- Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

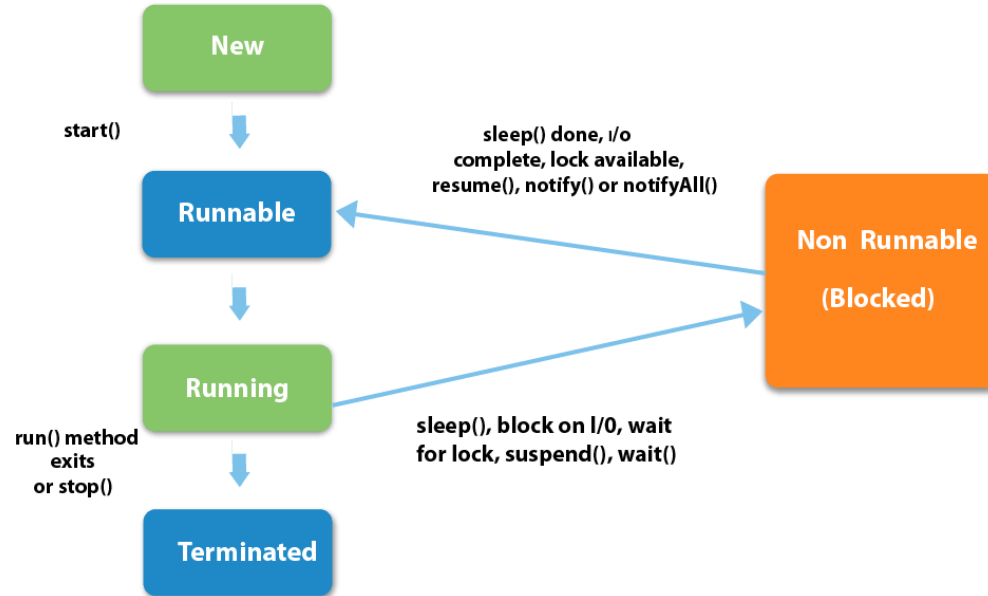
- Nhược điểm của đa luồng

- Càng nhiều luồng thì xử lý càng phức tạp.
- Xử lý vấn đề về tranh chấp bộ nhớ, đồng bộ dữ liệu khá phức tạp.
- Cần phát hiện tránh các luồng chết (dead lock), luồng chạy mà không làm gì trong ứng dụng cả.



- 1) Process-based Multitasking (Multiprocessing)
 - Each process has an address in memory. In other words, each process allocates a separate memory area.
 - A process is heavyweight.
 - Cost of communication between the process is high.
 - Switching from one process to another requires some time for saving and loading registers, memory maps, updating lists, etc.
- 2) Thread-based Multitasking (Multithreading)
 - Threads share the same address space.
 - A thread is lightweight.
 - Cost of communication between the thread is low.

- Lifecycle



- There are two ways to create a thread:
 - By extending Thread class
 - Override lại phương thức run ở lớp này
 - start() method of Thread class is used to start a newly created thread. It performs following tasks:
 - A new thread starts(with new callstack).
 - The thread moves from New state to the Runnable state.
 - When the thread gets a chance to execute, its target run() method will run.

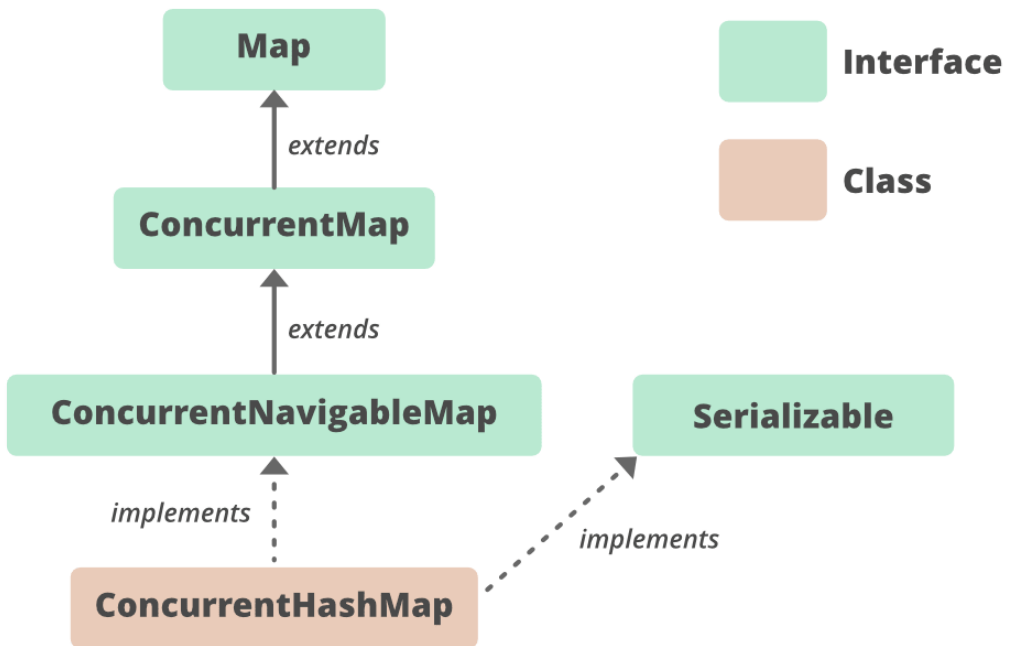
```
public class TheadSimple extends Thread {  
    public void run() {  
        System.out.println("thread is running...");  
    }  
  
    public static void main(String args[]) {  
        TheadSimple t1 = new TheadSimple();  
        t1.start();  
    }  
}
```

- By implementing Runnable interface.
 - have only one method named run()
 - Cách hay được sử dụng và được yêu thích là dùng interface Runnable, bởi vì nó không yêu cầu phải tạo một lớp kế thừa từ lớp Thread. Trong trường hợp ứng dụng thiết kế yêu cầu sử dụng đa kế thừa, chỉ có interface mới có thể giúp giải quyết vấn đề. Ngoài ra, Thread Pool rất hiệu quả và có thể được cài đặt, sử dụng rất hơn giản.

```
public class RunnableSimple implements Runnable {  
    public void run() {  
        System.out.println("thread is running...");  
    }  
  
    public static void main(String args[]) {  
        RunnableSimple runnable = new RunnableSimple();  
        Thread t1 = new Thread(runnable);  
        t1.start();  
    }  
}
```


- Thread safe
 - nhiều luồng được tạo từ cùng một biến đối tượng chia sẻ dữ liệu và điều này có thể dẫn đến sự không nhất quán dữ liệu khi các luồng được sử dụng để đọc và cập nhật dữ liệu được chia sẻ.
 - việc cập nhật bất kỳ giá trị trường nào là một quá trình, nó đòi hỏi ba bước; đầu tiên để đọc giá trị hiện tại, thứ hai để thực hiện các thao tác cần thiết để có được giá trị cập nhật và thứ ba để gán giá trị cập nhật cho tham chiếu trường.
 - <https://vncoder.vn/bai-viet/thread-safety-luong-an-toan-trong-java-va-dong-bo-hoa-synchronized-cac-luong-trong-java>
 - <https://toihoccodeblog.wordpress.com/tag/singleton-thread-safe/>

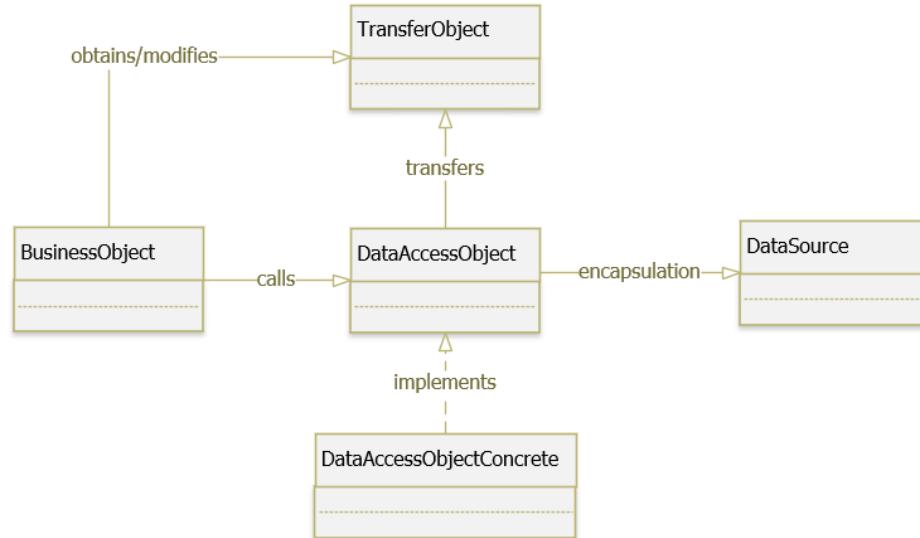
- Thread Safety trong java là quá trình làm cho chương trình của chúng ta an toàn để sử dụng trong môi trường đa luồng, có nhiều cách khác nhau để chúng ta có thể làm cho luồng chương trình của mình an toàn.
 - Đồng bộ hóa (Synchronization) là công cụ dễ dàng nhất và được sử dụng rộng rãi nhất cho Thread Safety trong java.
 - synchronized
 - khóa và mở khóa tài nguyên trước khi bất kỳ luồng nào đi vào mã được đồng bộ hóa, nó phải thu được khóa trên Object và khi việc thực thi mã kết thúc, nó sẽ mở khóa tài nguyên có thể bị khóa bởi các luồng khác. Trong khi đó, các luồng khác đang ở trạng thái chờ để khóa tài nguyên được đồng bộ hóa.
 - Sử dụng các lớp Atomic Wrapper từ gói `java.util.concurrent.atomic`. Ví dụ: `AtomicInteger`
 - Sử dụng các khóa từ gói `java.util.concurrent.locks`.
 - Sử dụng các lớp thread safe collection (bộ sưu tập luồng an toàn) , sử dụng `ConcurrentHashMap` để biết safe thread.
 - Sử dụng từ khóa “volatile” với các biến để làm cho mọi luồng đọc dữ liệu từ bộ nhớ, không đọc từ bộ đệm của luồng.



- **Data Access Object (DAO)** Pattern là một trong những Pattern thuộc nhóm cấu trúc (Structural Pattern).
 - DAO Pattern dựa trên các nguyên tắc thiết kế **abstraction** và **encapsulation**. Nó bảo vệ phần còn lại của ứng dụng khỏi mọi thay đổi trong lớp lưu trữ
 - Trong Java, DAO được triển khai theo nhiều cách khác nhau như Java Persistence API, Enterprise Java Bean (EJB), Object-relational mapping (ORM) với các implement cụ thể như Hibernate, iBATIS, Spring JPA, ...



- **BusinessObject** : đại diện cho Client, yêu cầu truy cập vào nguồn dữ liệu để lấy và lưu trữ dữ liệu.
- **DataAccessObject (DAO)**: là một interface định nghĩa các phương thức trừu tượng việc triển khai truy cập dữ liệu cơ bản cho BusinessObject để cho phép truy cập vào nguồn dữ liệu (DataSource).
- **DataAccessObjectConcrete** : cài đặt các phương thức được định nghĩa trong DAO, lớp này sẽ thao tác trực tiếp với nguồn dữ liệu (DataSource).
- **DataSource** : là nơi chứa dữ liệu, nó có thể là database, xml, json, text file, webservice, ...
- **TransferObject** : là một POJO (Plain old Java object) object, chứa các phương thức get/set được sử dụng để lưu trữ dữ liệu và được sử dụng trong DAO class.



- **Transfer Object/ Data Transfer Object Pattern** là một dạng Architectural Design Pattern, được sử dụng khi chúng ta muốn truyền dữ liệu qua lại giữa các tầng trong ứng dụng, giữa Client – Server.
 - Data Transfer Object (DTO) còn được gọi là **Value Object (VO)**.

