休斯顿华夏中文学校-Java Class Notes

- Google Classroom
- Java 8 API Document
- Java 16 API Document
- Useful icons and mermaid
- 编程入门该学哪门语言?

Table of Contents

Getting Start

Check Software Installation

```
java -version
javac -version
```

- here is a sample output:

```
C:\Users\12818\workspace\java>java -version
java version "16.0.2" 2021-07-20
Java(TM) SE Runtime Environment (build 16.0.2+7-67)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.2+7-67, mixed mode, sharing)

C:\Users\12818\workspace\java>javac -version
javac 16.0.2
```

❓What is Java?
✅ Java is a general-purpose, class-based, object-oriented programming language ~~designed for having lesser implementation dependencies~~. It is a computing platform for application development.

1. define data type
2. create object use the data type
3. use the object to do the job (OOP)

📌❗ Knowledge Base

> No class no Java.
> In order to run java application, you need JRE installed on your computer.
> In order to compile java source code, you need JDK installed on your computer.

❓ what is JVM?

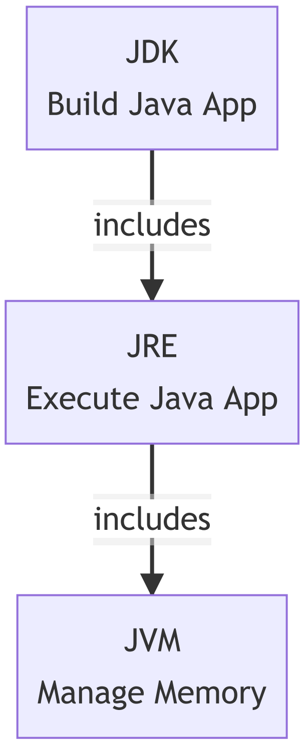
✅ The JVM stands for Java Virtual Machine, which manages system memory and provides a portable execution environment for Java-based applications

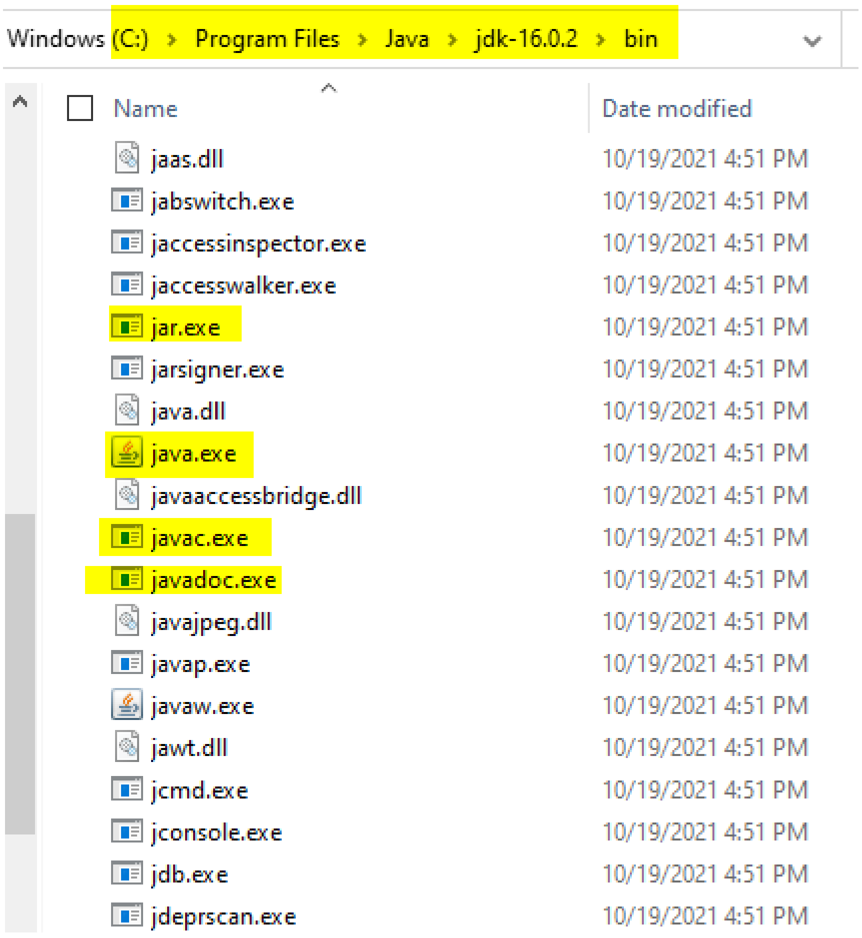
❓ What is JRE? (java.exe)

✅ The JRE stands for Java Runtime Environment, which provides the minimum requirements for executing a Java application. It includes JVM, core classes, and supporting files.

❓ What is JDK? (javac.exe)

✅ THe JDK stands for Java Development Kit, which is a software development environment used for developing Java applications and applets. It includes JRE, an interpreter/loader (java.exe), a compiler (javac.exe) and archiver (jar.exe), a documentation generator (javadoc.exe), and more...





Windows (C:) › Program Files › Java › jdk-16.0.2 › bin

Name	Date modified
jaas.dll	10/19/2021 4:51 PM
jabswitch.exe	10/19/2021 4:51 PM
jaccessinspector.exe	10/19/2021 4:51 PM
jaccesswalker.exe	10/19/2021 4:51 PM
jar.exe	10/19/2021 4:51 PM
jarsigner.exe	10/19/2021 4:51 PM
java.dll	10/19/2021 4:51 PM
java.exe	10/19/2021 4:51 PM
javaaccessbridge.dll	10/19/2021 4:51 PM
javac.exe	10/19/2021 4:51 PM
javadoc.exe	10/19/2021 4:51 PM
javajpeg.dll	10/19/2021 4:51 PM
javap.exe	10/19/2021 4:51 PM
javaw.exe	10/19/2021 4:51 PM
jawt.dll	10/19/2021 4:51 PM
jcmd.exe	10/19/2021 4:51 PM
jconsole.exe	10/19/2021 4:51 PM
jdb.exe	10/19/2021 4:51 PM
jdeprscan.exe	10/19/2021 4:51 PM

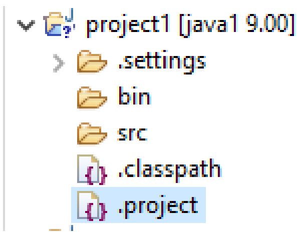
❓ What is java project in Eclipse?

✔ A java project in eclipse is a structured files system which include definition and configuration about the project, such as [.project] and [.classpath] files, [src], [bin] folder and more. the [src] folder is used for storing Java source code, and the [bin] folder is used for storing compiled bytecode files.

Online documentation and tutorial

❓ What is package?

✔ A package in Java is used to group related classes. Think of it as **a folder in a file directory**.



```
project1 [java1 9.00]
  > .settings
    bin
    src
    .classpath
    .project
```

```
1. Create Java Project: langbasic
2. Create Package: langbasic (by package name convention)
<com, org>.<company name>.<project name>.<package name>...
3. Create class: Hello.java
for any class name by convention, use Upercase for first letter
```

- My First Java Program
    1. java reserved keyword in purple color: package,public,class,static,void
    2. public and static is modifier which can be used to modify class, method, or variable
    3. class is used to declare a java class
    4. void is a method return data type, which means nothinbg to return
    5. package is actual file folders
    6. cannot use private modifier for class (inner class is ok.)
    7. default class can be used within same package
    8. package declaration line cannot be removed
    9. ❓how to rename class name: ✔highlight class name > right-click > refactor > Rename
    10. cannot remove public or static modifier and void return type from main() method.
    11. cannot change main() method name. JRE will use it to run the class.
    12. cannot change argument String[] in main() method, it is part of the signature of main().
    13. the args variable name can be changed to something else.
    14. ; cannot be removed, it is used to complete the java statement.
    15. "" define a String object.
    16. 😢System.out.println([String]), there are more than one signature for println() method, such as println([int]);

method signature:

1. method name
2. number of arguments
3. type of arguments
4. return data type(it may not belong to signature, but is part of method declaration)

static method

-

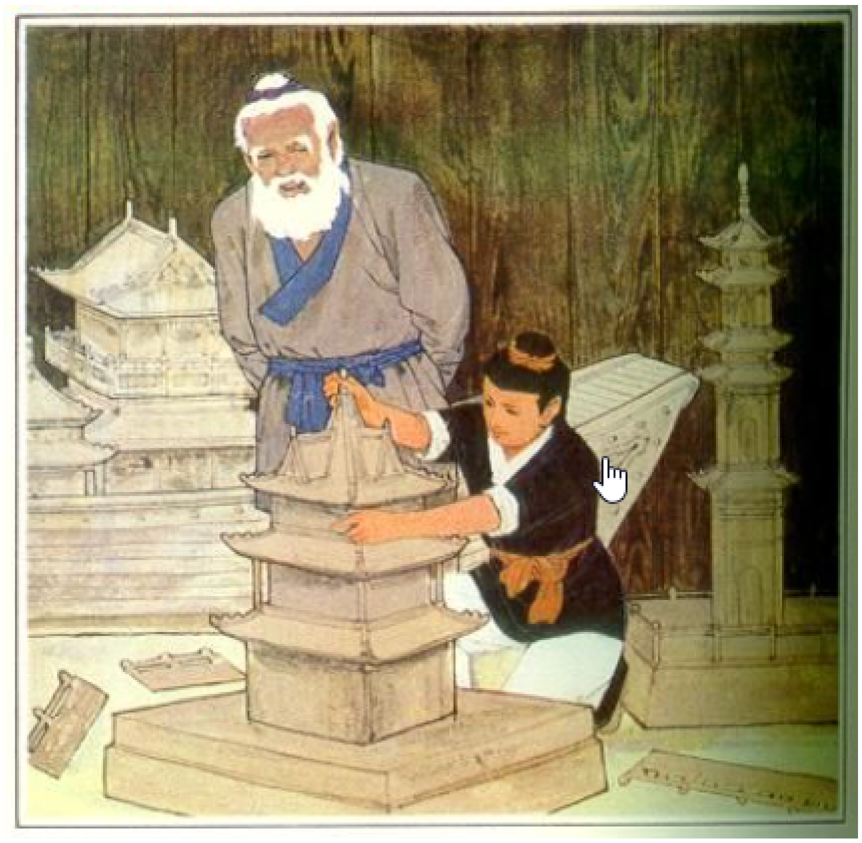
💡👉 Knowlodge Base

> 1. static method can be called by using the class name;
> 2. static meshod can be called by instance;
> 3. static method cannot use instance variables. (indipendent from instance)

❓ How to learn new computer language?
✔

1. do it by yourself

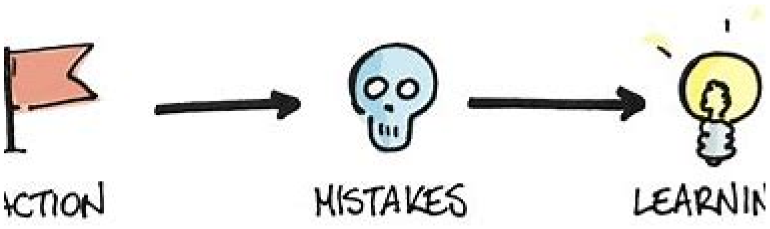
2. learn from mistakes
3. take good learning notes
4. programming practice
5. teach someone else



Lǔbān is a famous master carpenter of ancient China. **Lǔbān** studied hard for three years and learned all the skills. The old master want to try him more, and destroy all the models, let him build them all over again. He made one by one exactly the same as the original all based on his memory. The old master created a lot of new models for him to build. He pondered and did it, and the results were made according to the style of the master. The old master was very satisfied.

鲁班是著名的中国古代的木匠师傅。鲁班苦学了三年，把所有的手艺都学会了。老师傅还要试试他，把模型全部毁掉，让他重新造。他凭记忆，一件一件都造得跟原来的一模一样。老师傅又提出好多新模型让他造。他一边琢磨一边做，结果都按师傅说的式样做出来了。老师傅非常满意。

> Everything you have learned, you need do it all by yourself, until then, whatever you've learned indeed belongs to yourself.



Basic skills(questions and answers)

- ❓How do I create a Java project?
  ✅Right-Click on explore window ⟹ New ⟹ Java Project ⟹ enter project name
- ❓How do I create Java package?
  ✅Right-click src folder ⟹ New ⟹ package ⟹ enter package name
- ❓How do I cerate a Java class?
  ✅Right-click package name ⟹ New ⟹ class ⟹ enter class name
- ❓Hod wo I run a Java class?
  ✅Click the green run button on toolbar
- ❓Can I rename the main() method?
  ✅No❗❗❗
- ❓How can I save the class file?

```
✅there are more than one way to save the file:
1. Ctrl+s
2. File ⟹ Save
3. File ⟹ Save All
4. click save button on toolbar
```

File Structure

Organize Class Notes File Structure

```
<java>
    ├── ReadMe.md
    ├── doc/
    |     ├── images/
    |     ├── homeworks/
    |     |       ├── mardown01.md
    |     |       └── mardown02.md
    |     ├── eclipseTrics.md
    |     ├── markdownTrics.md
    |     ├── utilities.md
    |     └── java.md
    ├── langbasic/bin/ (byte code)
    └── langbasic/src/ (source code)
```

Homework 1

Homework 2

My First Java Program

```
1. Create Java Project: javaclass
2. Create Package: com.huaxia.javaclass (by package name convention)
<com, org>.<company name>.<project name>.<package name>...
3. Create class: Hello.java
for any class name by convention, use Upercase for first letter
```

- My First Java Program
    - System.out.println()
    - // single line comment
    - every Java statement line ends with ;
    - public before the class is not important
    - main() method must be public
    - main() method must be static
    - void on main() method is return type
    - main() method name is special, change it will cause main() not found error.
    - String[] is part of main() method signature, cannot be changed
    - args is variable name which can be changed.
    1. java reserved keyword in purple color: package,public,class,static,void
    2. public and static is modifier which can be used to modify class, method, or variable
    3. class is used to declare a java class
    4. void is a method return data type, which means nothinbg to return
    5. package is actual file folders
    6. cannot use private modifier for class

7. default class can be used within same package
8. package declaration line cannot be removed
9. ❓how to rename class name: ✔️highlight class name > right-click > refactor > Rename
10. cannot remove public or static modifier and void return type from main() method.
11. cannot change main() method name. JRE will use it to run the class.
12. cannot change argument String[] in main() method, it is part of the signature of main().
13. the args variable name can be changed to something else.
14. ; cannot be removed, it is used to complete the java statement.
15. "" define a String object.
16. 😢System.out.println([String]), there are more than one signature for println() method, such as println([int]);

> method signature:

1. method name
2. number of arguments
3. type of arguments
4. return data type(it may not belong to signature, but is part of method declaration)

Getting farmiliar with your keyboard

Keyboard definition

Print

- Print.java

1. System.out.printf(), is a formatted print out.
2. %d placeholder for integer
3. \n new line character
4. // single line comment
5. /** ... */ multilines comment
6. System.out.println()
7. System.out.print()
8. System.out.printf()

❓ What are the differences between print(), println(), and printf() methods?
✔️ (homework)

- Homeworks

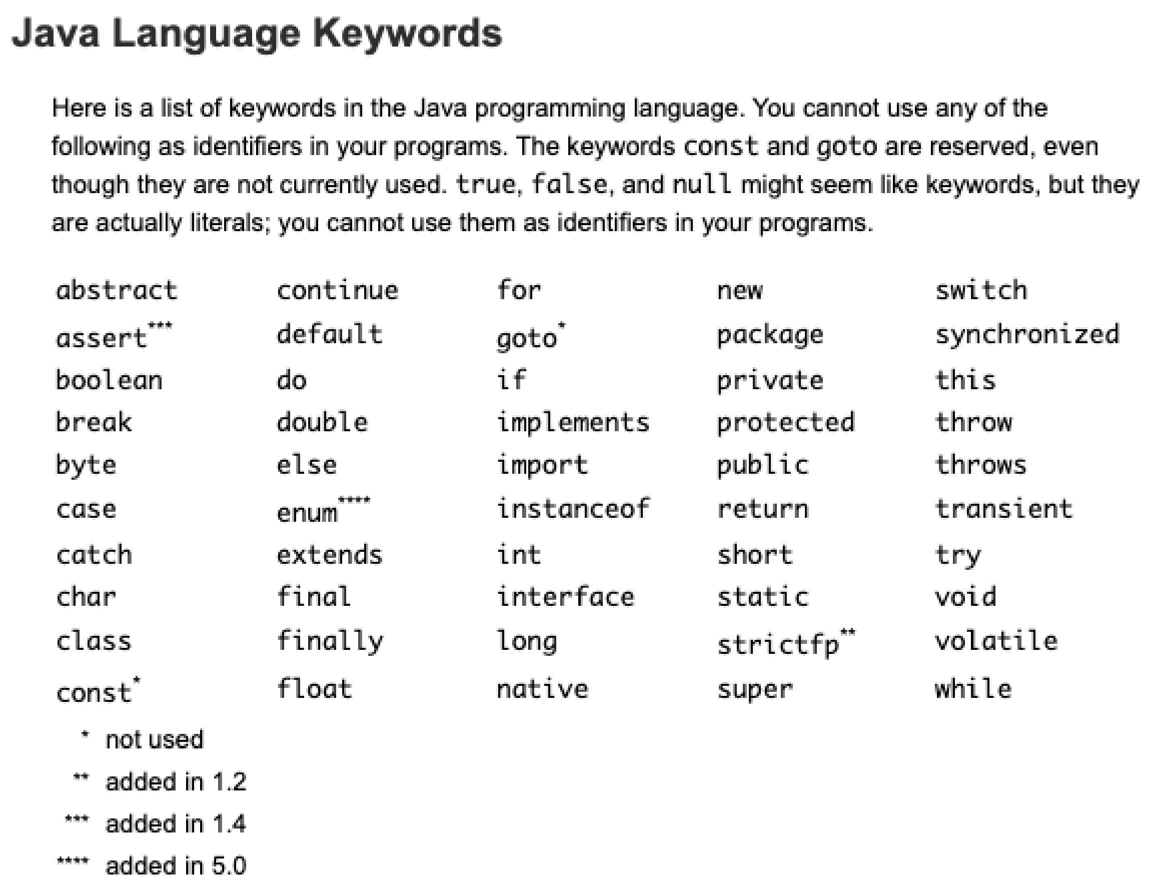
1. Math Competition
2. variable01
3. variable02

Variable

❓ Why using variable?
✔ we can define variable once, and use it at many place, so that you only make less change when you need change the value.

naming rules

1. any variable name cannot start with number.
2. variable name can be combination of letters and numbers $, _, a->z, A->Z, 0->9, no other special characters
3. cannot use Java reservered keywords as variable name.
4. cannot define same variable name within same code scope.
5. you can re-assign new value on same variable without declaration.



Java Language Keywords

Here is a list of keywords in the Java programming language. You cannot use any of the following as identifiers in your programs. The keywords `const` and `goto` are reserved, even though they are not currently used. `true`, `false`, and `null` might seem like keywords, but they are actually literals; you cannot use them as identifiers in your programs.

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

*	not used
**	added in 1.2
***	added in 1.4
****	added in 5.0

define variable, assign value

📌❗ Knowlodge Base

> **the naming rules above is also apply for class name, method name.**

Comment

1. single line comment
2. multiple line comment
3. for document
4. block code from executing

Scanner

❓ What is Scanner?
✅ is Java built-in class which allow program to get user input from console.

Scanner

1. System.in is the console
2. **new** is a Java keyword used to create instance of a class.
3. nextLine(), nextInt(), ..., find all other functions from Java API document👇 below.

Java API Document
seach for java.util.Scanner

📌❗ Knowlodge Base

> ❗ public class name must to be the same as the java file name!

❓can we run java programs on command prompt?
✅ Yes.

1. make sure your JDK java compiler installed on your machine correctly.
2. make sure your JDK bin path is set on your system path.

```
C:\Program Files\Java\jdk-15.0.1\bin
javac.exe: java compiler which compiles java source code to byte code(.class)
java.exe:  JRE, Java runtime environment
```

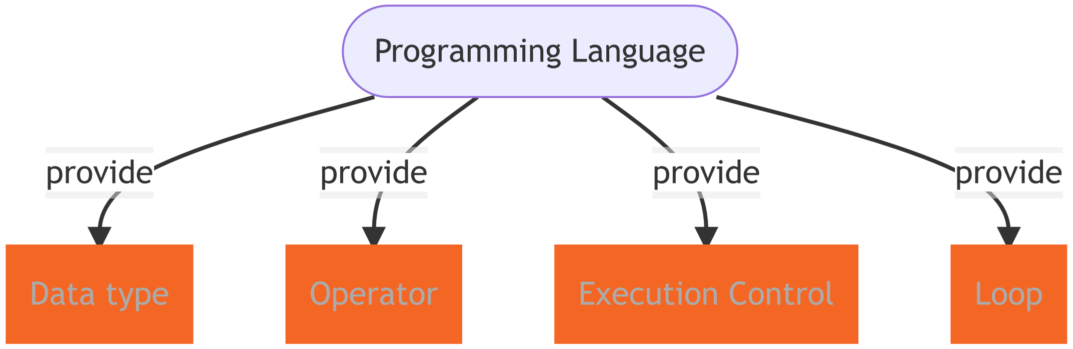
3. compile your source by using

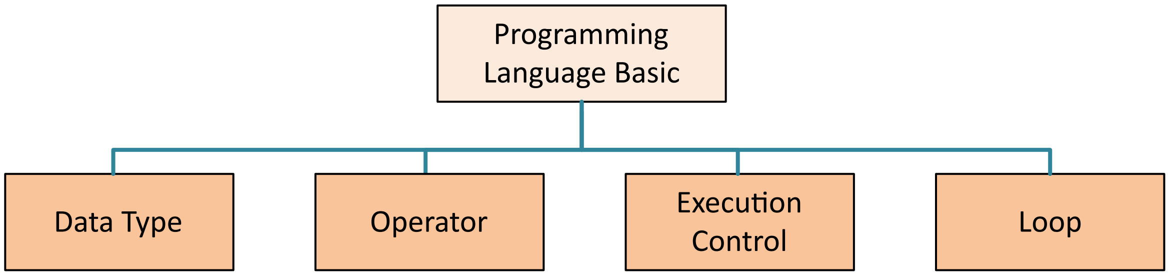
```
javac <java file name include .jave extention>
```

4. Run your byte code by using JRE

```
java <class name>
```

Language Basics





Data Type

- DataType.java
  - primitive data type (boolean, byte, char, short, int, long, float, double)

> cast: small value can assign to large container; must cast large value before assign to small container.

- Java built in data type (String,)
- User defined data type

1. primitive data type (boolean, byte, short, int, long, float, double)
2. Java defined data type (String, StringBuffer, ...)
3. Developer defined data type

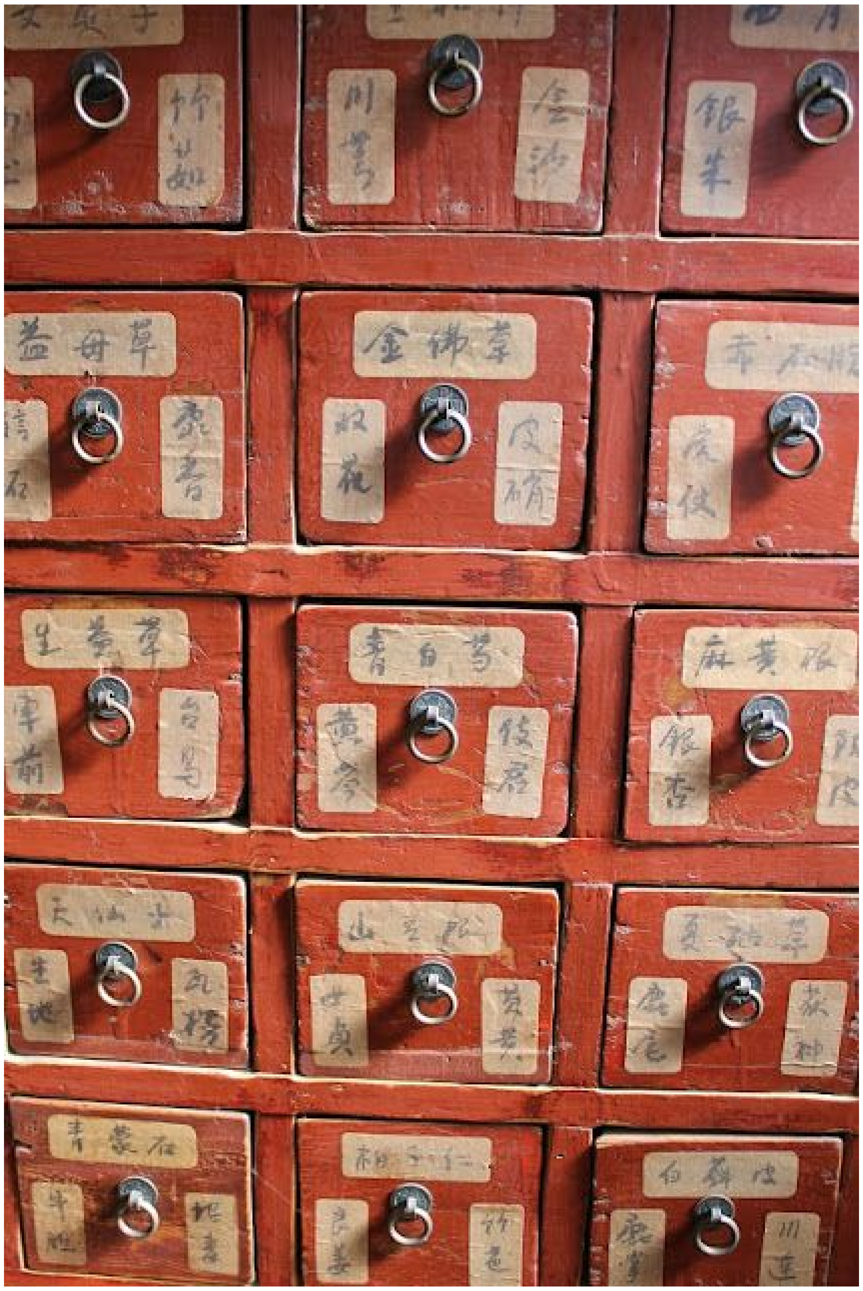
> every class extends(inherits) from **java.lang.Object** which is the root class for all java classes
> you can override toString() method to represent different object
> if you create your own constructor, the default constructor no longer available

❓ What is differences between Primitive and Java defined data type?

> ✔ 1. java defined data type has a lot built in methods can be used.
> 1. java defined data type you need use **new** keyword to create an instance.

📌❗ **Knowlodge Base**

> Java programming is nothing but create and use data type.



$$2^7\times0+2^6\times1+2^5\times0+2^4\times0+2^3\times0+2^2\times0+2^1\times0+2^0\times1=65$$

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

Array

- ArrayTester.java
  - int[] primitive array
  - String[] Java defined array
  - Cup[] Developer defined array
  - Arrays.toString() method
-

ArrayList

- ArrayList Test
  👎😥 Arrays.asList(iList) return Arrays$ArrayList which cannot use add() operation.
  ✔️ iList = new ArrayList(iList); // then iList is a ArrayList.

Operator

❓ What are operators for java language?

- Operators.java

  - Arithmatic operator: +,-,*,/,%
  - compound assignment operator: +=, -=, *=, /=, %=
  - binary operator: ++, --,
  - comparison operator: >, <, >=, <=, ==, !=
  - logical operator: && and, || or, ! not
  - ternary operator: a<b?a:b
  - bitwise operator: &, |, ^
  - precendence: *, /; +, -; ()

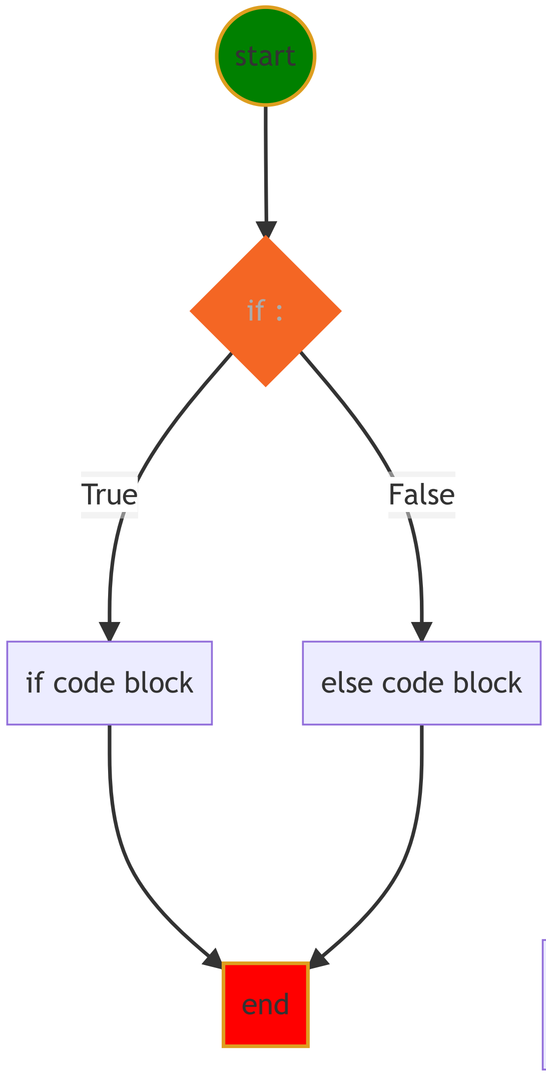
    ```
    &: bitwise and
    |: bitwise or
    ^: bitwise xor
    ~: bitwise compliment
    ```

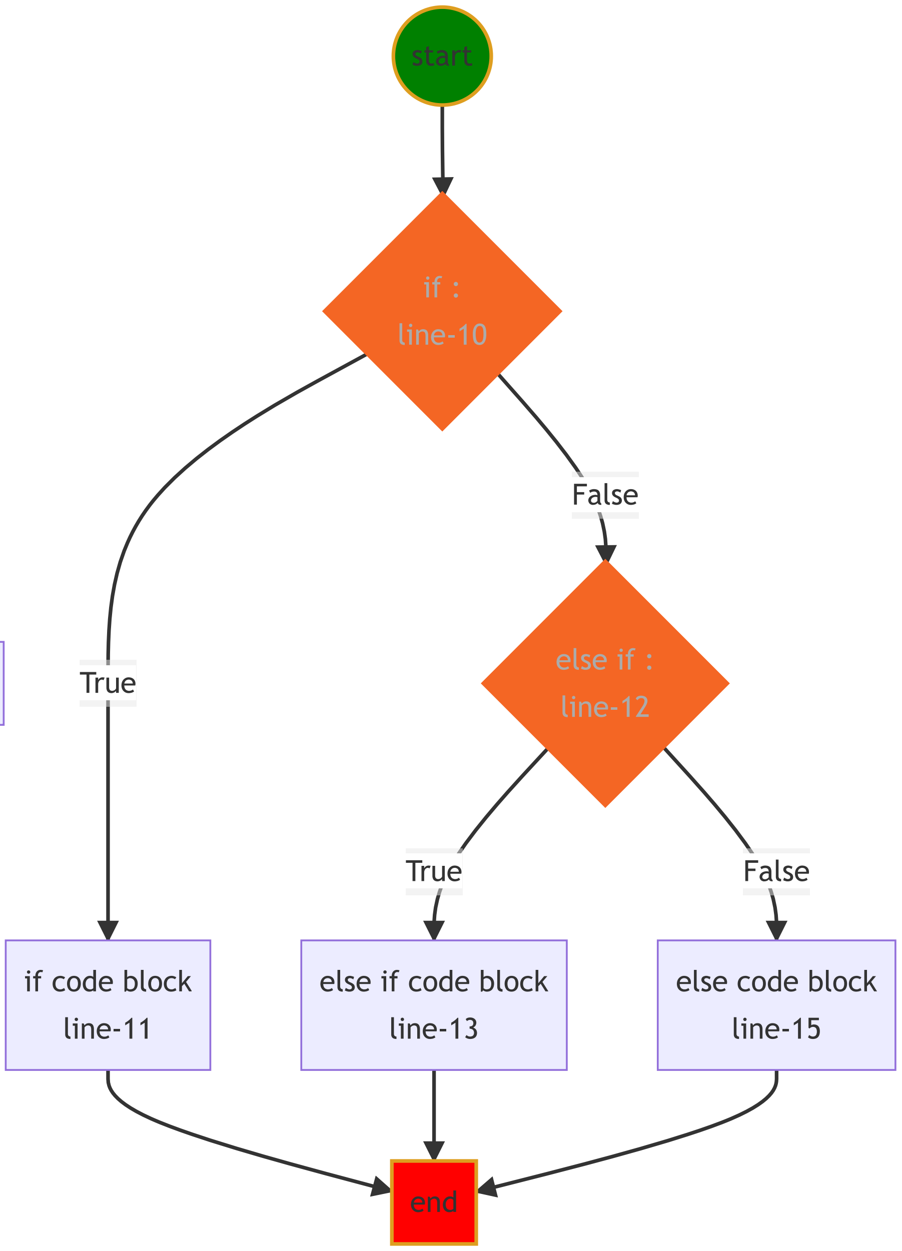
- Homeworks
  1. quiz02.md
  2. operator01.md
  3. operator02.md

Excution Control

if-else

Execution control





- if-else syntax

switch

[switch]

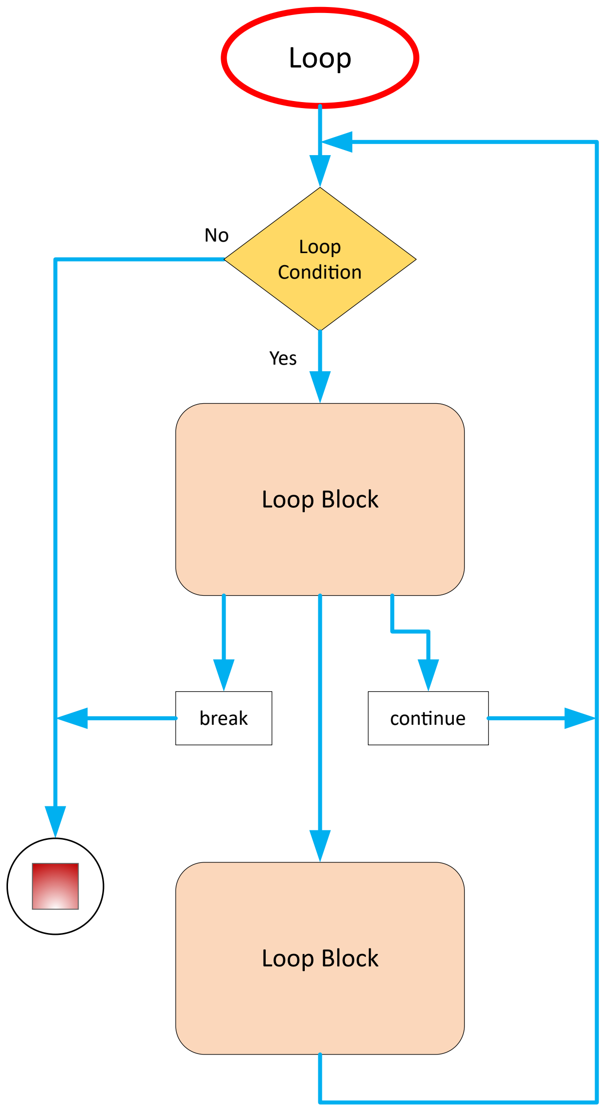
- Homeworks

Loop

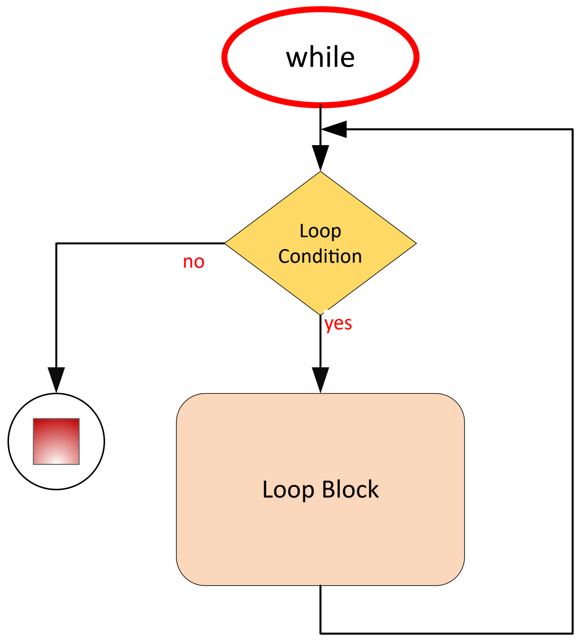
- For loop

- for loop (1. initial index; 2. loop condition; 3. adjust index), **continue**, **break**
- for each loop

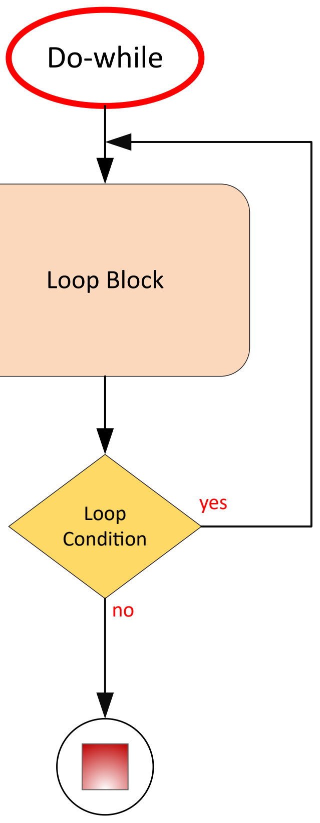
> go through each element in the iterable variables such as array, ArrayList, ...



- for/while loop/do-while
- While loop



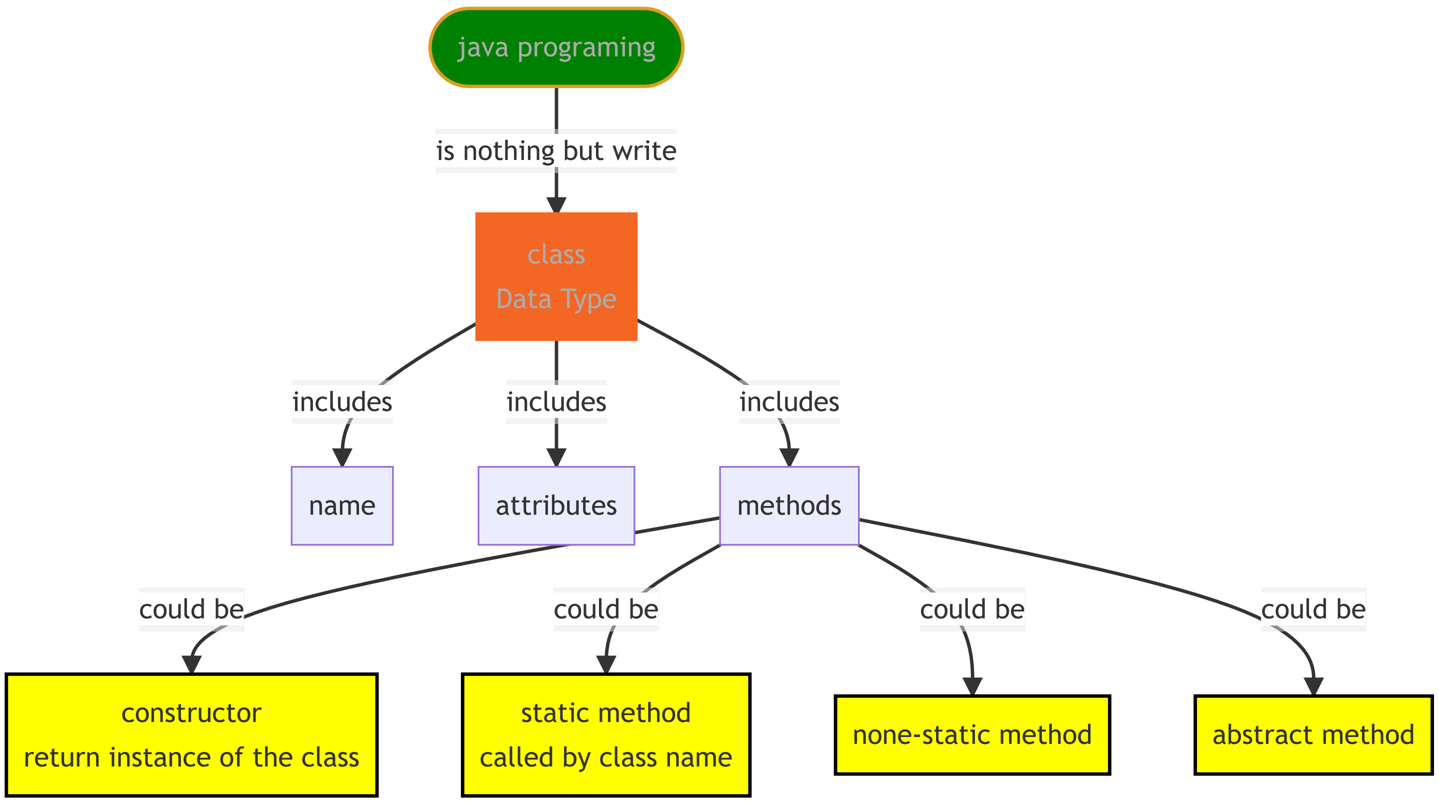
- do-while loop



```
while loop has 3 part:
1. initialize variable, a=0
2. variable condition, a<10
3. adjust variable, a +=1
```

❓ What is **static** method?

✅ a static method can be called by the class name, the method has nothing to do with the class attributes.



Java class

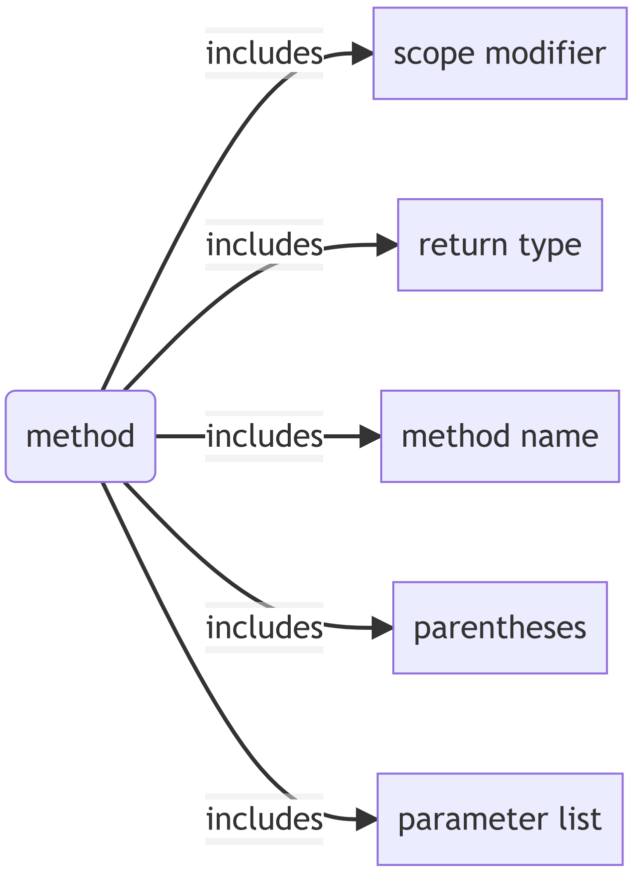
❓ What is java class?
✅ Java class is a software developer defined code blocks as a **data type** can be used to create an instance of that class. It includes class name, attributes and methods.

❓ What is constructor?
✅ the class constructor is special method which returns an instance of that class.

1. Constructor must use class name as its name;
2. the constructor can be modified by modifiers such as: private, public, default;
3. the constructor has no return data type, since it retuens the instance of this class;
4. there can be more than one constructors with different signatures so called method overloading;

❓ What is method signature?

When we define a method which includes the following parts



✅ includes 1. method name; 2. parameter data type; 3. number of parameters;

📌❗ **Knowledge Base**
💡👉 may concern a return type also as part of the signature;
💡👉 in one class there is only one method signature allowed. or cause compiler error.

❓ Parameter vs. argument
✅

> An **argument** is a value passed to a function when the function is called. An argument when passed with a function replaces with those variables which were used during the function;
> In other hand, a **parameter** is a variable used to define a particular value during a function definition. In the example below, a and b are **parameters**.

params vs. argument

```
public class Example {

        public static int multiply(int a, int b)
        {
                return a + b;
        }

        public static void main(String[] args)
        {
                int x = 2;
                int y = 5;

                // the variables x and y are arguments
                int sum = multiply(x, y);

                System.out.println("SUM IS: " + sum);
        }
}
```

where x, y are the arguments.

❓ How do I set arguments to main() method in eclipse?
✔ Right-Click class name > Properties > Run/Debug Settings > Click class name > Edit > Arguments > enter argument list > Apply and Close

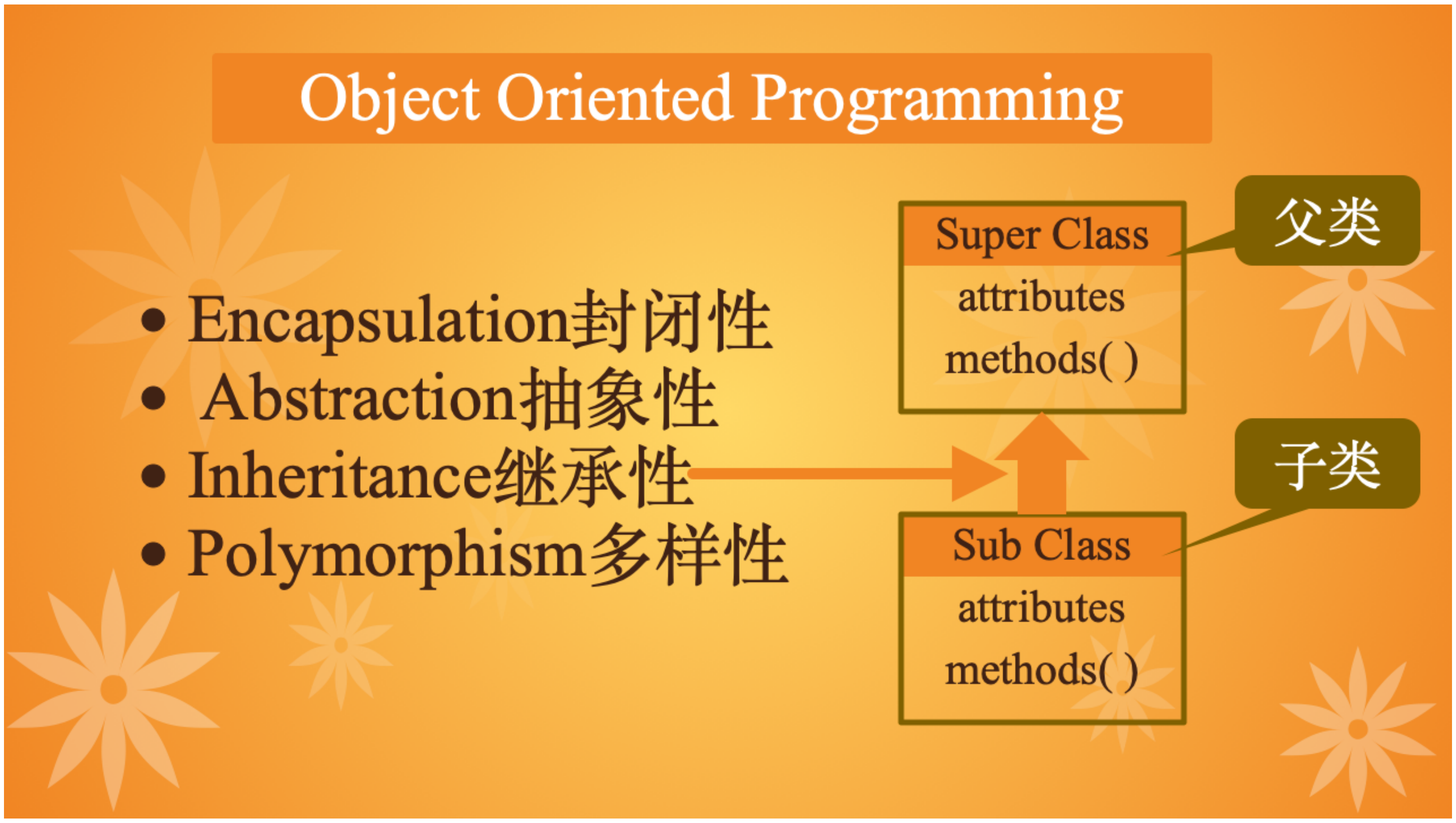
❓ How do I call java program from DOS command window?
✔

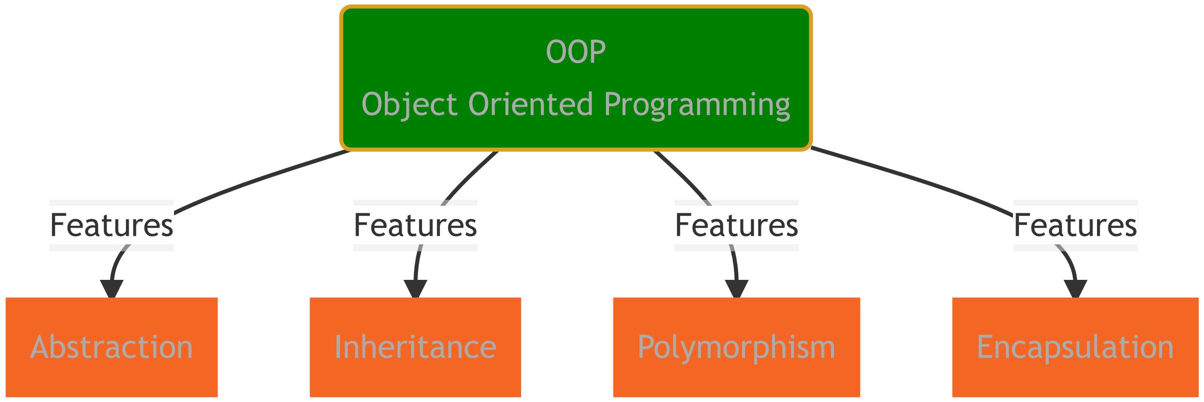
1. Open DOS window
2. change directory to the classes(bin) folder
3. run java command on DOS window

```
C:\Users\12818\workspace\java\huaxia\bin>java com.huaxia.java1.Example1 4 5
[4, 5]
4 x 5 = 20.000000
Hello, John
```

OOP

❓ What is OOP?
✔ Object Oriented Programming concept





Abstraction

❓ What is **Abstraction**?
✅ Abstract object attributes and functionalities in the real world that only interested in the software to define a java class.

Student.java
Circle.java

❓ What is toString() method in java?
✅ toString() is a method defined in **Object** class which returns a String representation of the object. Developer can always override it in different return.

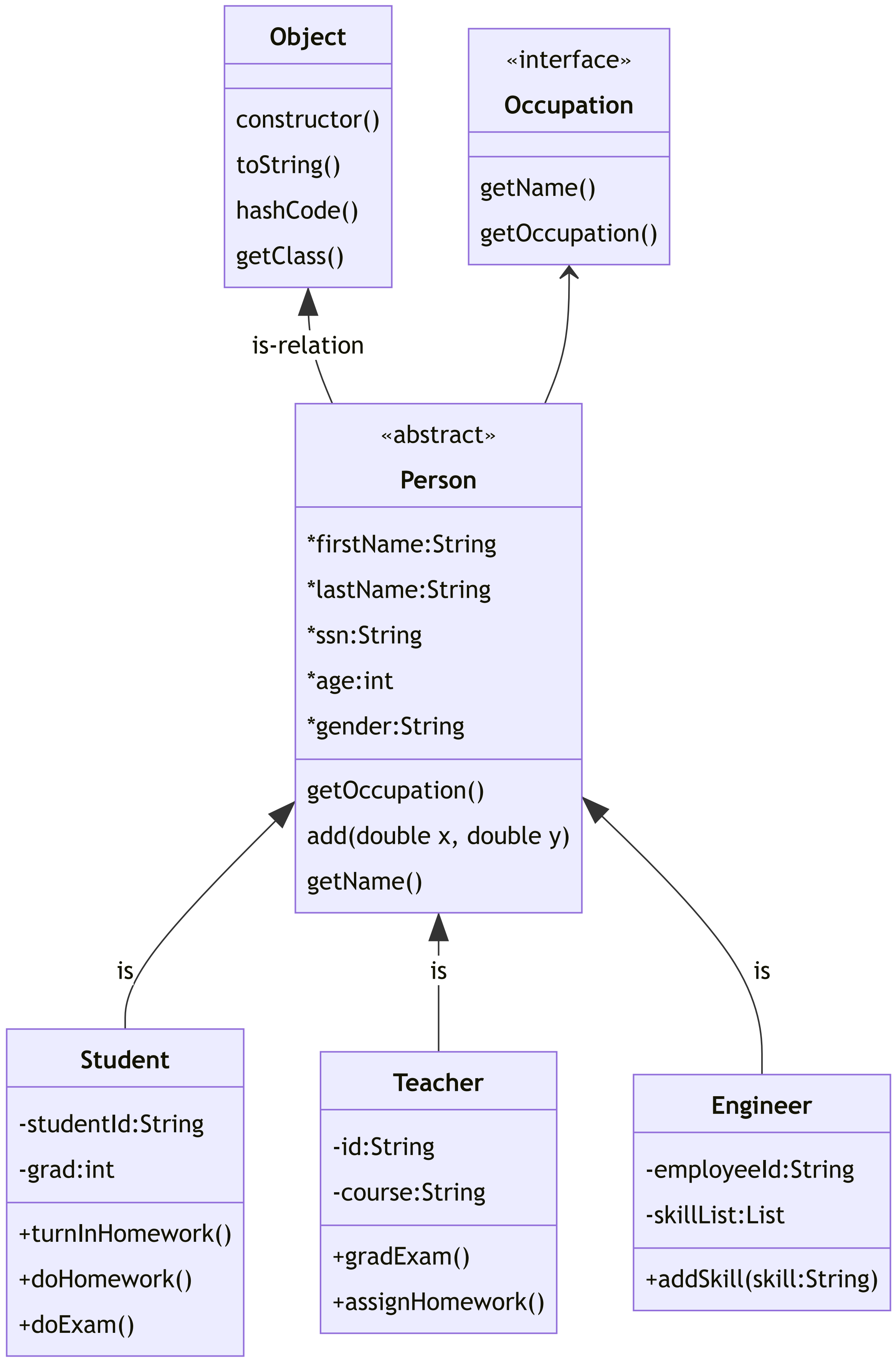
❓ What is **Object** class in java

✅ Class Object is the root of the class hierarchy. Every class has Object as a superclass.

Inheritance

❓ What is **Inheritance**?

✅ Subclass inherits features(attributes and methods) from superclass. (***is*** relationship)



Engineer.java

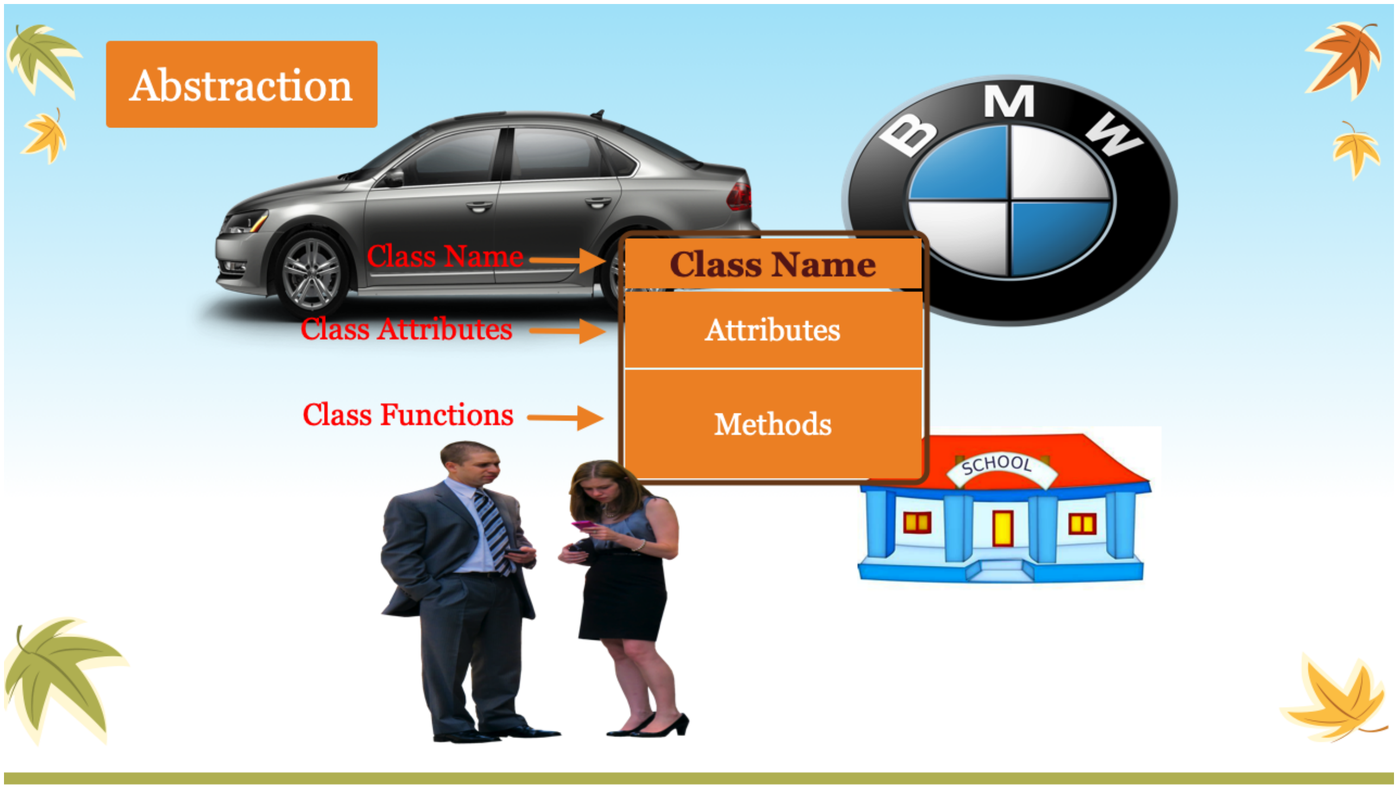
Test.java

❓ How to check **is** relation in Java?

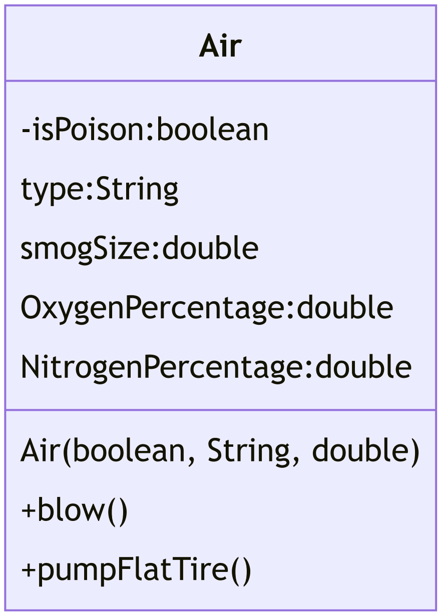
✔️ instanceof keyword

```
if(s1 instanceof Person){
 ...
}
```

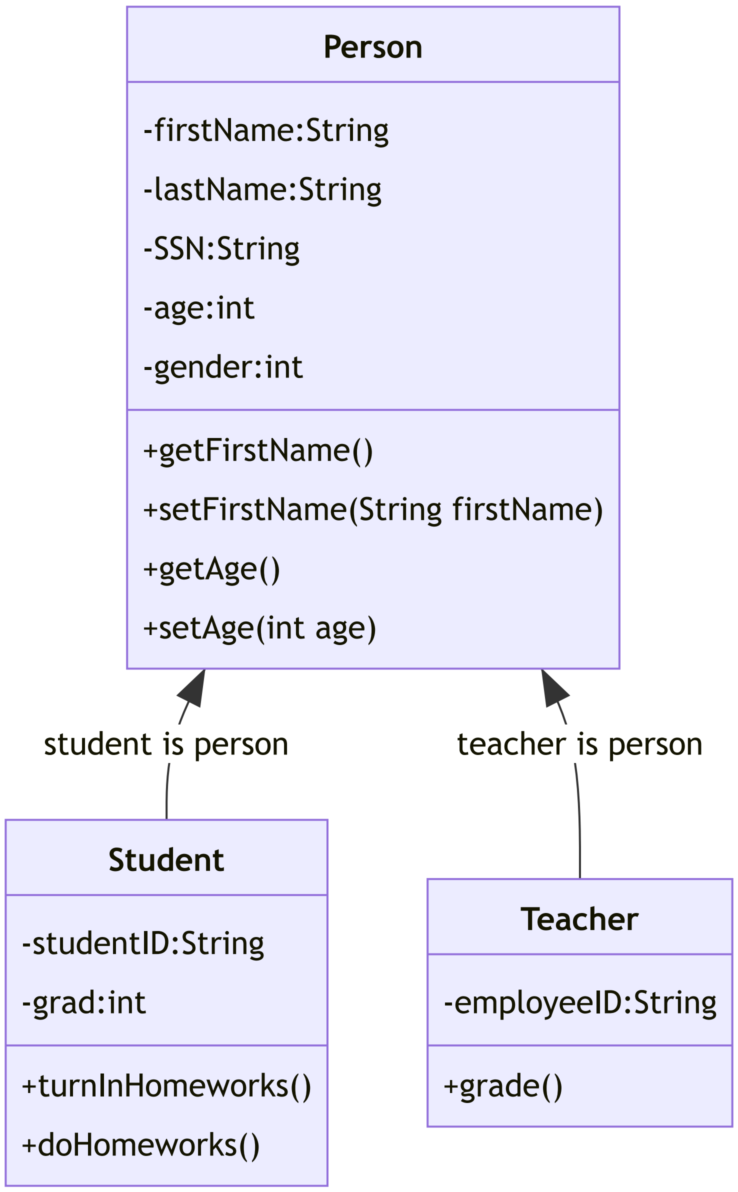
2. Abstraction: abstract object in the real world to write a class.



First class



3. Inheritance: subclass inherits features from super class.



Student is a Person, the relationship between Student and Person is **is relation**.
where the Student class is subclass of Person class, we call the Person as Superclass of Student class.

- Person, Superclass
- Student, Subclass of Person
- Teacher, Subclass of Person

Polymorphism

Polymorphism:

> allows a single task to be performed in different ways.
> Polymorphism in Java is the ability of an object to take many forms, it allows us to perform the same action in many different ways. (method Overriding, Overloading, runtime polymorphism)
> give different answer for the same question from different classes which inherit from same superclas or interface. (异类同功)

Test Polymorphism

Encapsulations

to make our java code isolated from unexpected changes. Encapsulation: private, protected

❓What is private modifier?

> ✔️private modifier make variable or method in the class can only be used within the class, which makes encapsulation possible.

class

❓What functions defined in Object class which are useful for us?
✔️the functions available in Object are

1. default constructor
2. toString()

❓What is construtor?
✔️Constructor is used to create an instance of the class.

Construtor

1. Constructor looks like a method which does NOT have return type since it always return the instance.
2. Constructor can use public, package, private and protected modifier
3. 😢👎If you defined your own constructor with arguments, the default constructor no longer works
   - ✔️create a default constructor which does **NOT** have any aruments.
   - ✔️add arguments when you call the constructor
4. 👌You can define many different constructors which has different signature.
5. 😢Subclass can NOT use Superclass constructor.
6. 😄Subclass can use public or protected methods defined in the superclass.
7. 😢👎👎👎class defined in the same package can call protected method.

8. 👍protected method cannot be called from different package.

❓What is protected modifier?
✅protected modifier allow subclass to access the superclass attributes or methods. protect from using by other classes.

getter, setter, toString, default constructor

❓How do I create getter/setter?
✅
❓How do I override toString() method?
✅
❓Why I want to override toString()?
✅

Inhritance

- Inherit from class
  Super class: Person
  Subclass Student
  Subclass Teacher
  - class can only inherit from one class, Java does NOT allow multiple inheritance
- Implements from Interface

❓What is interface?

👎 An interface is a completely "abstract class" that is used to group related methods with empty bodies:

✅👍 An interface is a special java code block that defined collection of abstract methods without implementation, and possible constants.

1. All methods defined in interface are ***abstract***.
2. All methods defined in interface are ***public***.
3. All vriables defined in interface are ***static***, ***final*** constants.
4. One class can implements more than one interface
5. to ***implements*** an interface, all methods defined in the interface need to be implemented (provide method body).

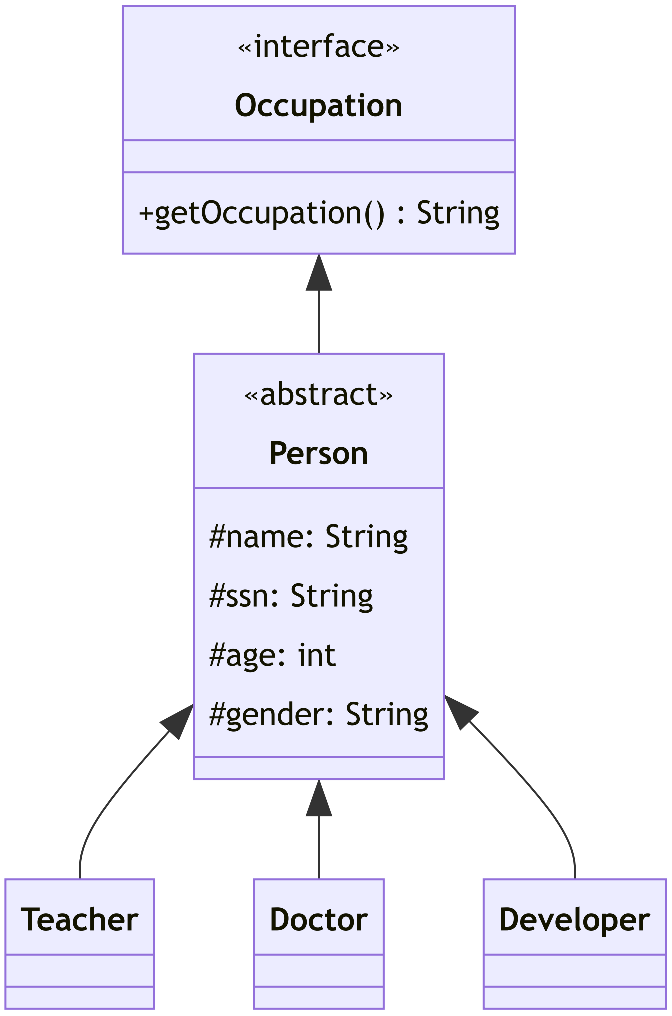
6. 😄An interface cannot be used to create an object.
7. Interface can be used to declare variable as a data type.

❓ How do I create interface?

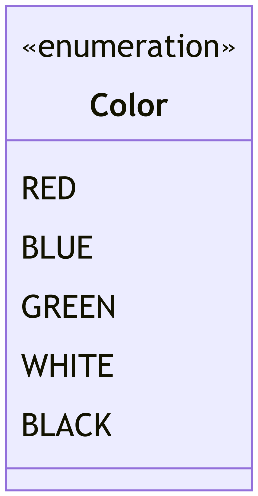
✅

❓How do I use the interface?

✅implements the interface in the class



- Annotations on classes
    1. << interface >>
    2. << abstract >>
    3. << Service >>
    4. << enumeration >>



interface

❓ What is interface?

✅ the following definition

❌👎Bad definition: An interface is a completely "abstract class" that is used to group related methods with empty bodies.

> ✅👍 An interface is a special java code block that define abstract methods, and possible constants.

❓ What is an abstract methond?

✅ a method with an ***abstract*** modifier with all method signatures without implementation.

1. All methods defined in interface are ***abstract***.
2. All methods defined in interface are ***public***.
3. All vriables defined in interface are ***static***, ***final*** constants.
4. All classes implement the interface need implement all methods defined in the interface.
5. An interface can not be used to create object.
6. An interface can be used to delear variable or method return type.

abstract class

❓ What is an abstract class?
✅ Abstract classes are similar to interfaces. You cannot instantiate them, and they may contain a mix

of methods declared with or without an implementation. Abstract class: is a restricted class 💡that cannot be used to create objects (to access it, it must be inherited from another class).

1. the abstract class can leave method as abstract. (can have abstract methods)
2. An abstract class can not used to create object.
3. if the abstract class implements an abstract method defined in the interface, its subclass won't be necessary to implement that method.

concrete class

❓ What is an concrete class?
✔️

1. A concrete class can be used to create an object.
2. A concrete class cannot include abstract method.
3. use **abstract** modifier to make the class abstract which allow no implementation of the abstract method defined in interface.

- Inherit from Abstract class
  1. need implements all abstract methods or make itself to be abstract

❓What is abstract method
Abstract method: can only be used in an abstract class, and it does not have a body. The body is provided by the subclass (inherited from).

❓Which should you use, abstract classes or interfaces?

> ✔️👇Consider using abstract classes if any of these statements apply to your situation:

- You want to share code among several closely related classes.
- You expect that classes that extend your abstract class have many common methods or fields, or require access modifiers other than public (such as protected and private).
- You want to declare non-static or non-final fields. This enables you to define methods that can access and modify the state of the object to which they belong.

> ✔️👇Consider using interfaces if any of these statements apply to your situation:

- You expect that unrelated classes would implement your interface. For example, the interfaces Comparable and Cloneable are implemented by many unrelated classes.
- You want to specify the behavior of a particular data type, but not concerned about who implements its behavior.

- You want to take advantage of multiple inheritance of type.

(🔥polymorphism) 👇see sample code below👇.

Person.java

Test.java

Method Overloading vs. Overriding

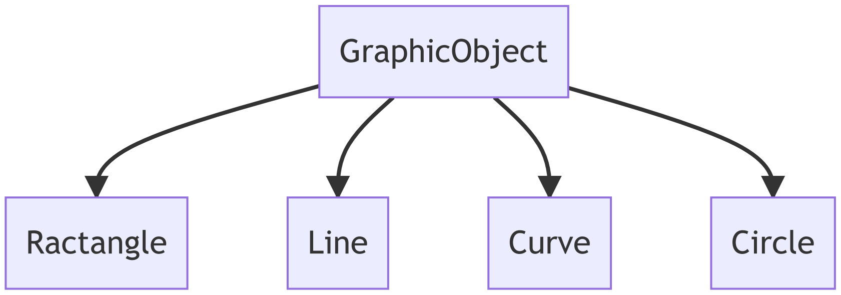
- Overloading:

> same method name within a class with different signature.

- Overriding:

> same method name and signature defined in subclass

An Abstract Class Example



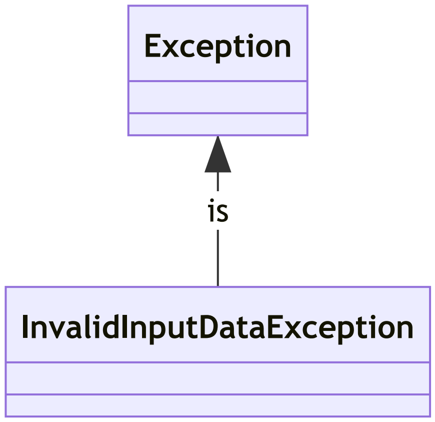
```
//AbstractSum.java
//abstract class
abstract class Sum{
   /* These two are abstract methods, the child class
    * must implement these methods
    */
   public abstract int sumOfTwo(int n1, int n2);
   public abstract int sumOfThree(int n1, int n2, int n3);

   //Regular method
   public void disp(){
        System.out.println("Method of class Sum");
   }
}

//ConcreteSum.java
//Regular class extends abstract class
class Demo extends Sum{

   /* If I don't provide the implementation of these two methods, the
    * program will throw compilation error.
    */
   public int sumOfTwo(int num1, int num2){
          return num1+num2;
   }
   public int sumOfThree(int num1, int num2, int num3){
          return num1+num2+num3;
   }
   public static void main(String args[]){
      Sum obj = new Demo();
      System.out.println(obj.sumOfTwo(3, 7));
      System.out.println(obj.sumOfThree(4, 3, 19));
      obj.disp();
    }
}
```

Exception

- calculate Circle Area without throwing Exception
  Circle area with radius=-2

❓ What is Exception?

✅ In java, Exception is a root class for all Exceptions.



- Using Exception root class in general cases
-

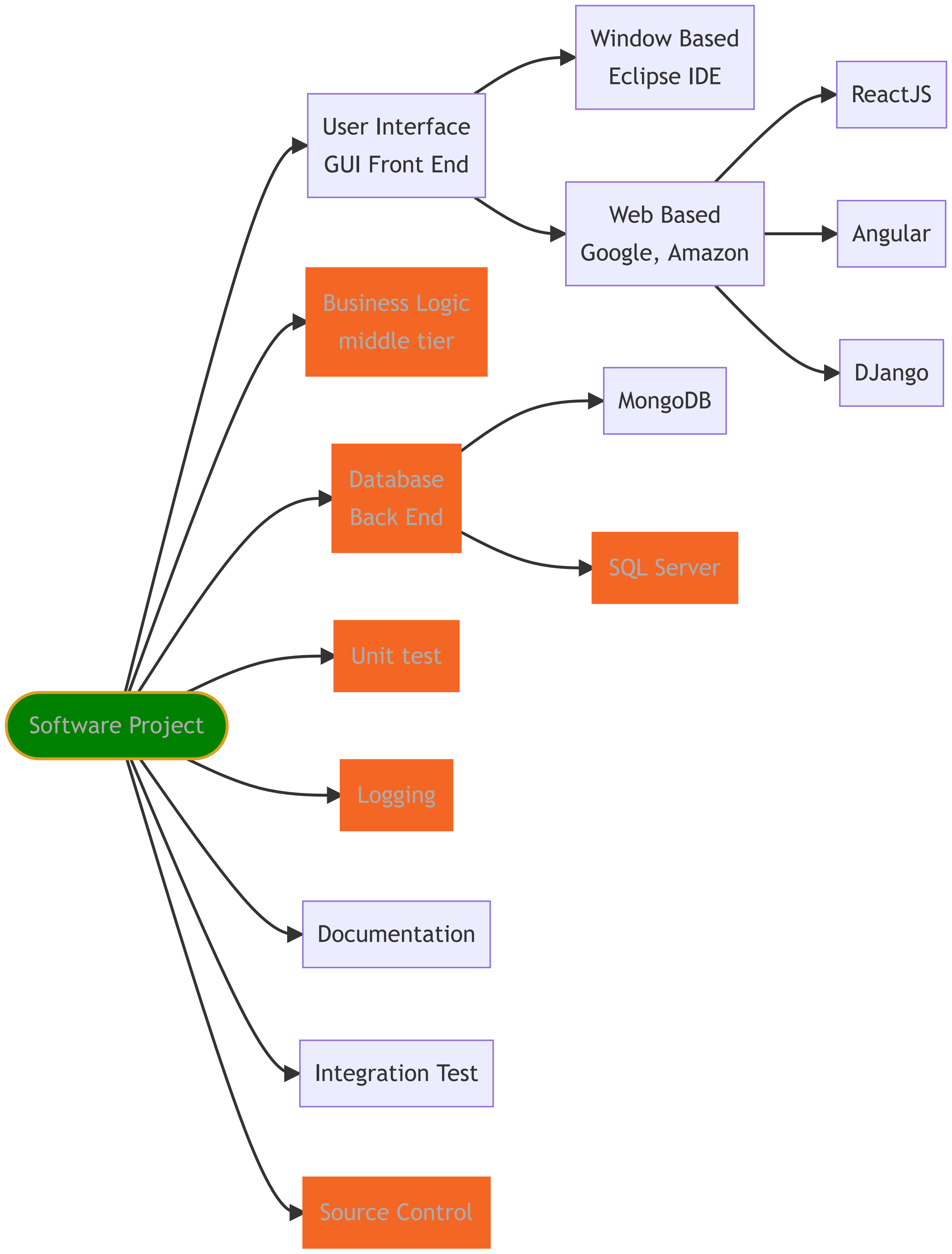
SimpleMath.div()

- Create your own specific Exception
  Create my own Exception
- try-catch block

❓ throw Exception is good enough for java app?

✅ No. most of times it is too late.

Parts of Softwere development

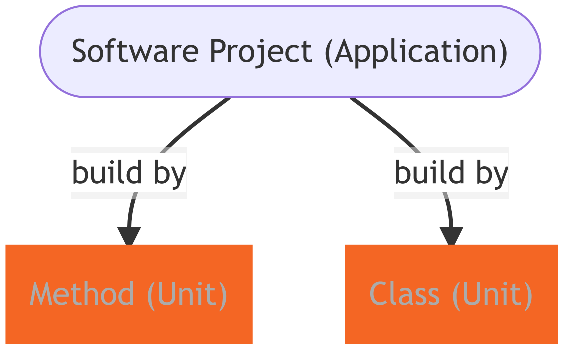


👎😢 throw Exception can help developer locate the problem, and root reason which causes the issues, but sometimes it is too late.

Unit test

❓What is Unit Test?

> ✔A unit is a specific piece of code to be tested, such as a function or a class. Unit tests are then other pieces of code that specifically exercise the code unit with a full range of different inputs, including boundary and edge cases.

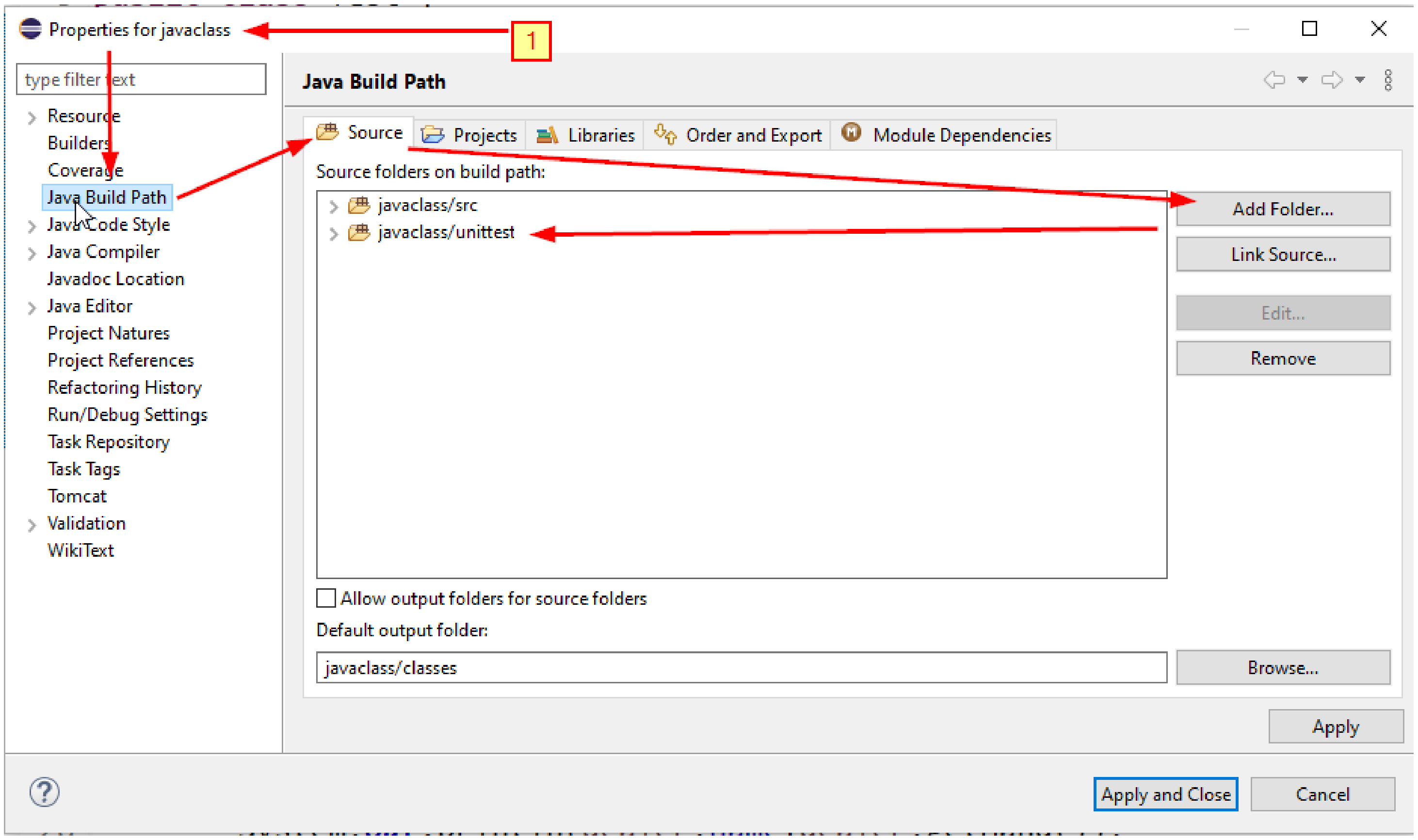


In order to make the project or application work well, we need make sure each small unit works correctly.

Simple math Unittest

TDD (Test Driving Development)

1. create new folder under project folder: unittest
2. add unittest folder to the source
   right-click project name ==> Properties ==> Java Build Path ==> Source (tab) ==> Add Folder (button) ==> unittest folder to source.



Properties for javaclass

1

type filter text

- Resource
- Builders
- Coverage
- Java Build Path
- Java Code Style
- Java Compiler
- Javadoc Location
- Java Editor
- Project Natures
- Project References
- Refactoring History
- Run/Debug Settings
- Task Repository
- Task Tags
- Tomcat
- Validation
- WikiText

Java Build Path

Source | Projects | Libraries | Order and Export | Module Dependencies

Source folders on build path:

- javaclass/src
- javaclass/unittest

Add Folder... | Link Source... | Edit... | Remove

☐ Allow output folders for source folders

Default output folder:

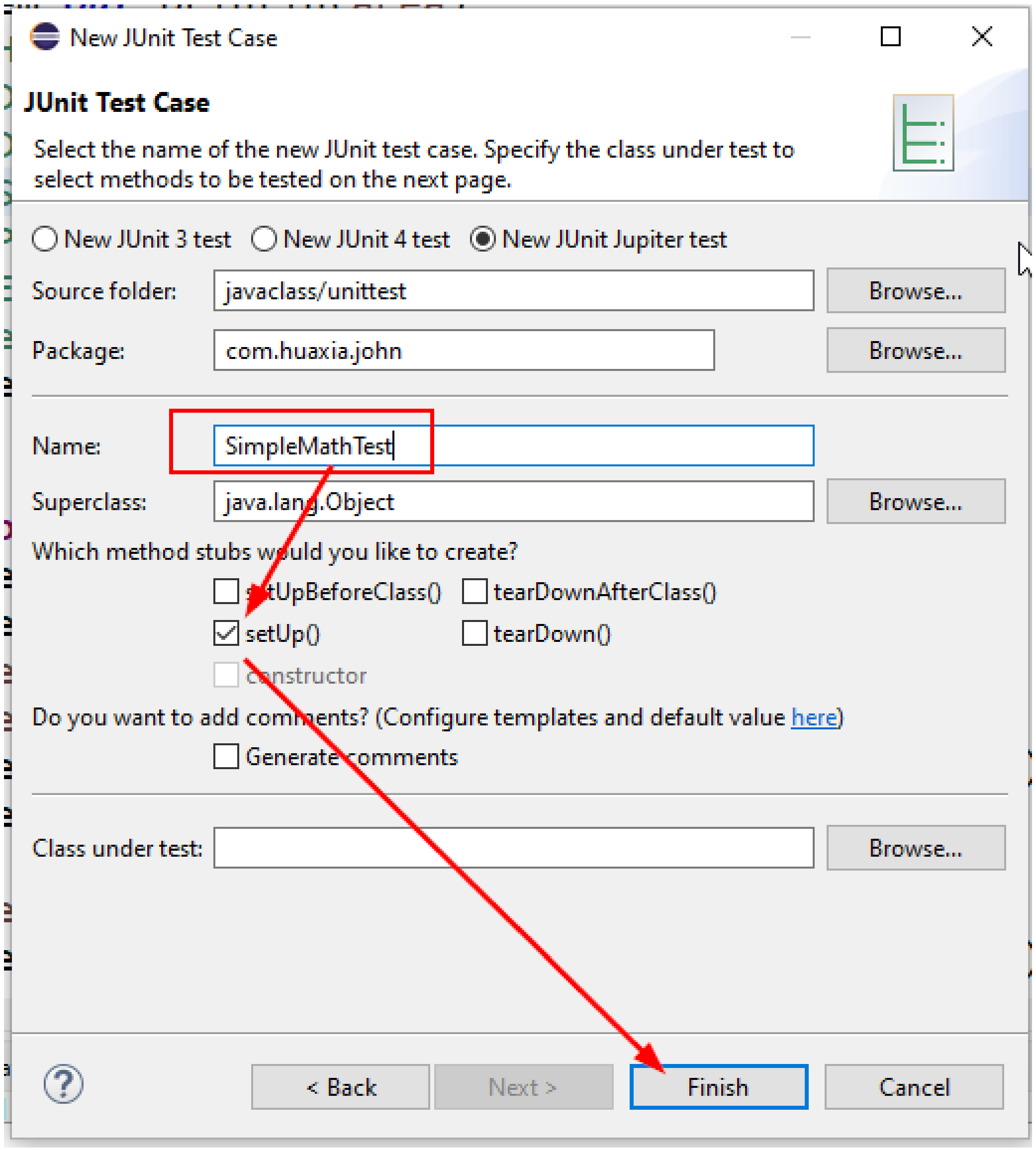
javaclass/classes

Browse...

Apply

Apply and Close | Cancel

3. create a new package: com.huaxia.javaclass
4. right-click package name ==> New ==> JUnit Test Case ==> enter the class name ==> finish(button)


New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. Specify the class under test to select methods to be tested on the next page.

New JUnit 3 test [✓ New JUnit Jupiter test] New JUnit 4 test

Source folder: javaclass/unittest

Package: com.huaxia.john

Name: SimpleMathTest

Superclass: java.lang.Object

Which method stubs would you like to create?

- [ ] setUpBeforeClass()
- [ ] tearDownAfterClass()
- [x] setUp()
- [ ] tearDown()
- [ ] constructor

Do you want to add comments? (Configure templates and default value here)

- [ ] Generate comments

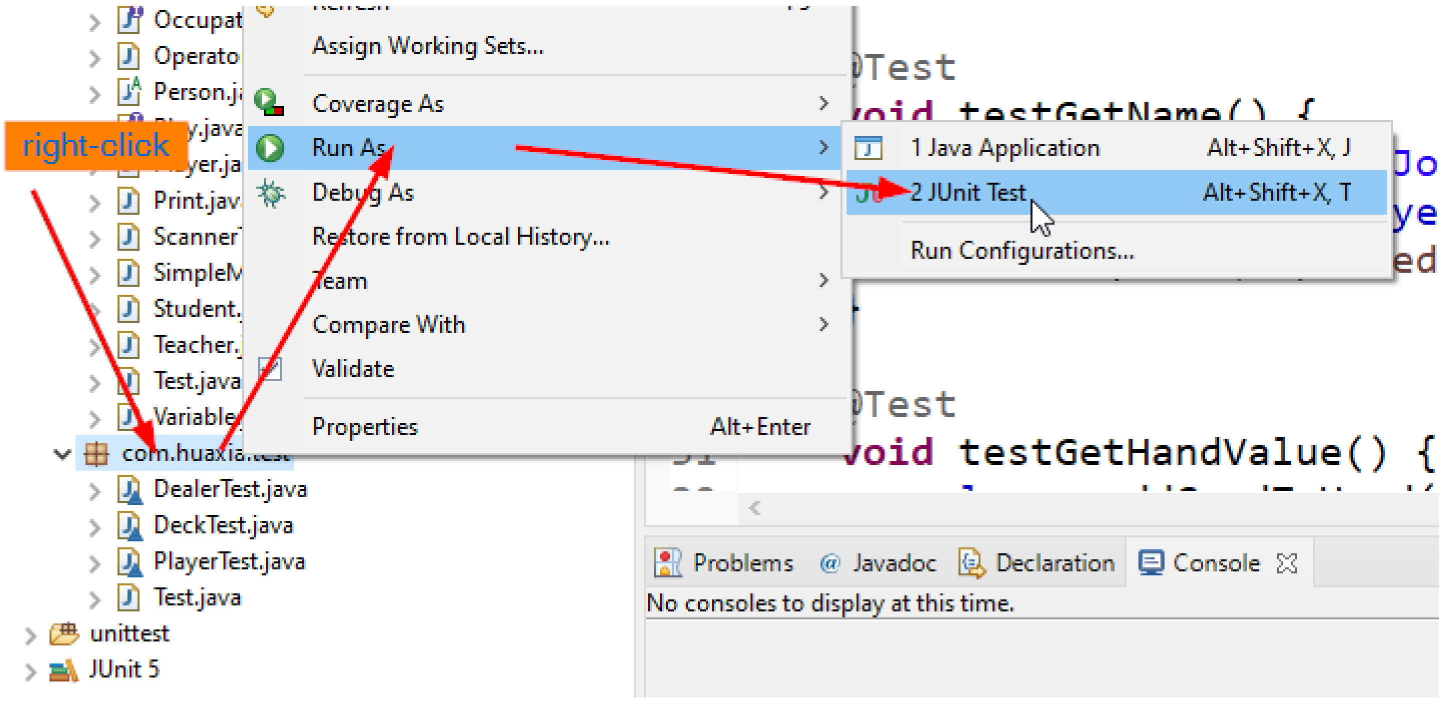
Class under test:

❓ How to run all unit test in once?

✔ Right-click unittest package name ==> Run As ==> JUnit Test or,

Highlight unittest package name ==> Run(menu) ==> Run As ==> JUnit Test or,

Highlight unittest package name ==> Alt+shift+x, t



No consoles to display at this time.

TDD (Test Driving Development)

❓ What is TDD?
✔

Logging

❓What is logging in programming?
✔Logging is keeping a record of all data input, processes, data output, and final results in a program to a file or database.
❓Why we need logging?
✔There are multiple reasons why we may need to capture the application activity.

1. Recording unusual circumstances or errors that may be happening in the program
2. Getting the info about whats going in the application

console log message
log to File
console log message
log to File

Blackjack Card Game

- Black Jack Rules

1. Object of the game:

> beat the dealer by getting a count as close to 21 as possible, without going over 21

2. Card Values

> ace is worth 1 or 11, J,Q,K are 10, other card is its pip value

3. Betting

> for simplicity, we don't bet.

4. Shuffle and cut

> the dealer shuffles the pack of card, no need player cut

5. Deal

> dealer gives one card face up to each player, and one card face up for himself. Another round of cards is then dealt face up to each player, but the dealer takes the second card face down.

6. Naturals

> If a player's first two cards are an ace and a "ten-card" (a picture card or 10), giving a count of 21 in two cards, this is a natural or "blackjack." If any player has a natural and the dealer does not, the dealer lose. If the dealer has a natural, other doesn't, dealer win. If both dealer and player have natural, no body wins.

7. The Play

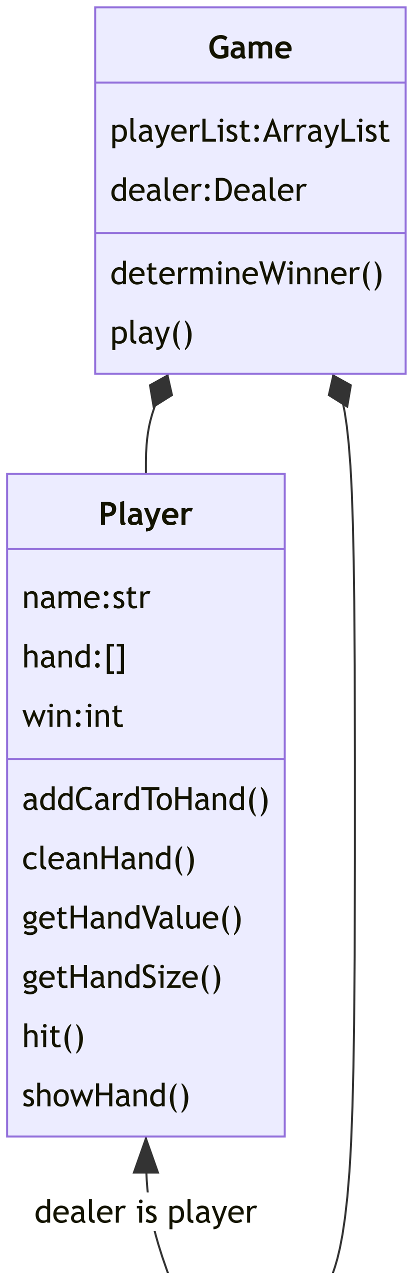
> any player on his turn must decide whether to "stand" (not ask for another card) or "hit" (ask for another card in an attempt to get closer to a count of 21, or even hit 21 exactly). Thus, a player may stand on the two cards originally dealt to them, or they may ask the dealer for additional cards, one at a time, until deciding to stand on the total (if it is 21 or under), or goes "bust" (if it is over 21). In the latter case, play loses the game. The dealer then turns to the next player and serves them in the same manner. The combination of an ace with a card other than a ten-card is known as a "soft hand," because the player can count the ace as a 1 or 11, and either draw cards or not. For example with a "soft 17" (an ace and a 6), the total is 7 or 17. While a count of 17 is a good hand, the player may wish to draw for a higher total. If the draw creates a bust hand by counting the ace as an 11, the player simply counts the ace as a 1 and continues playing by standing or "hitting" (asking the dealer for additional cards, one at a time).

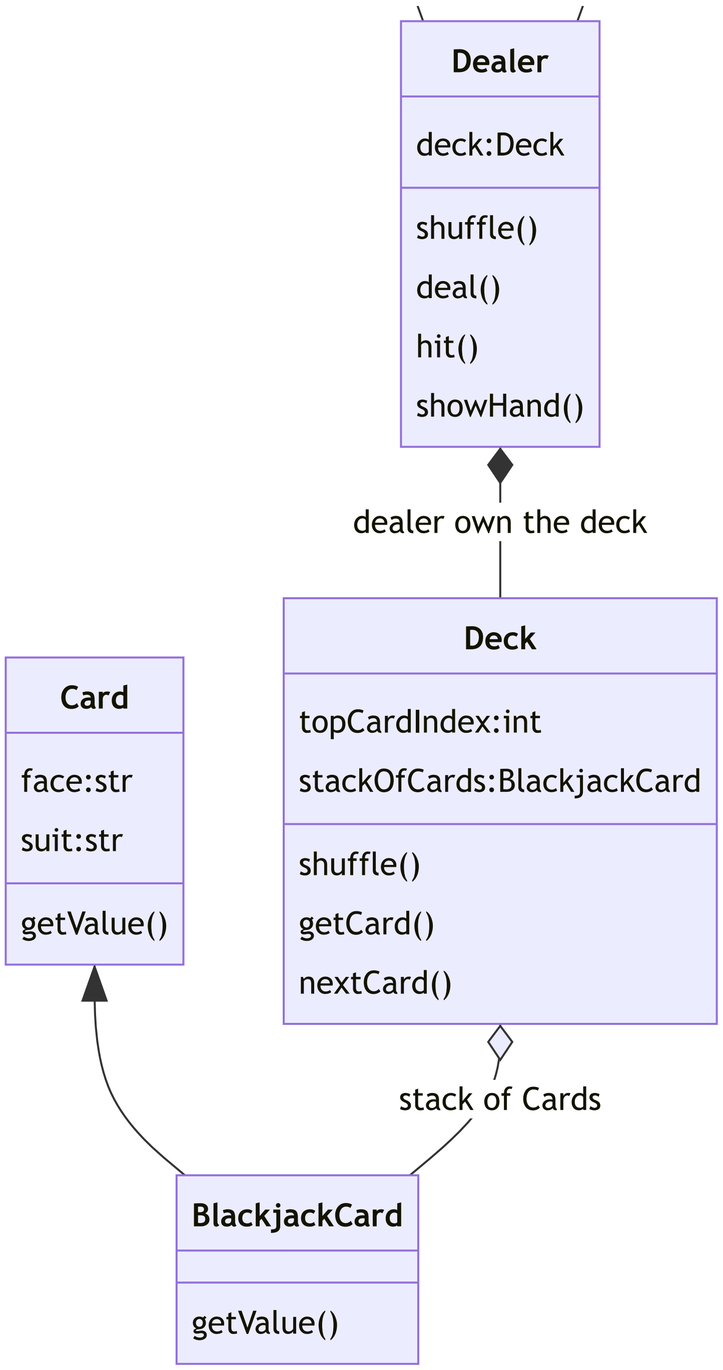
8. The Dealer's Play

> When the dealer has served every player, the dealers face-down card is turned up. If the total is 17 or more, it must stand. If the total is 16 or under, they must take a card. The dealer must continue to take cards until the total is 17 or more, at which point the dealer must stand. If the dealer has an ace, and counting it as 11 would bring the total to 17 or more (but not over 21), the dealer must count the ace as 11 and stand. The dealer's decisions, then, are automatic on all plays, whereas the player always has the option of taking one or more cards.

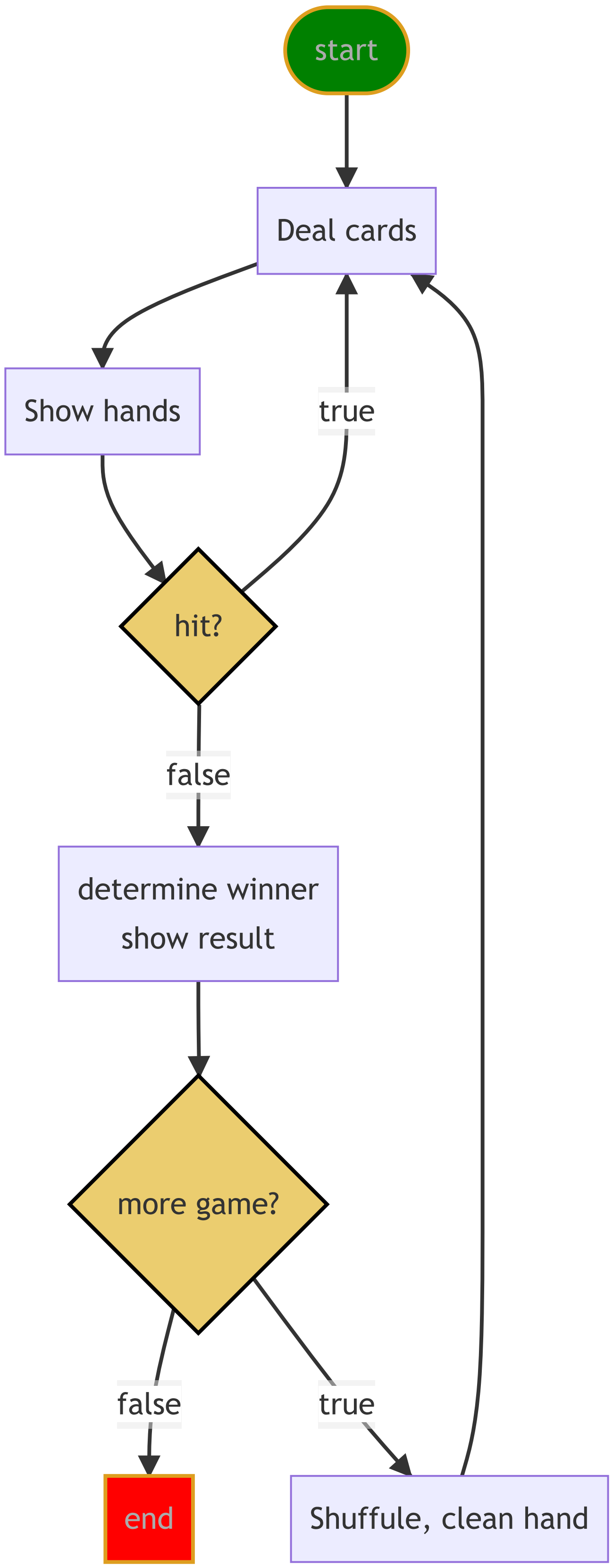
9. No Splitting Pairs
10. No Doubing Down
11. No Insurance
12. Reshuffling when start new game.

Object relationship

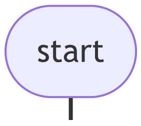


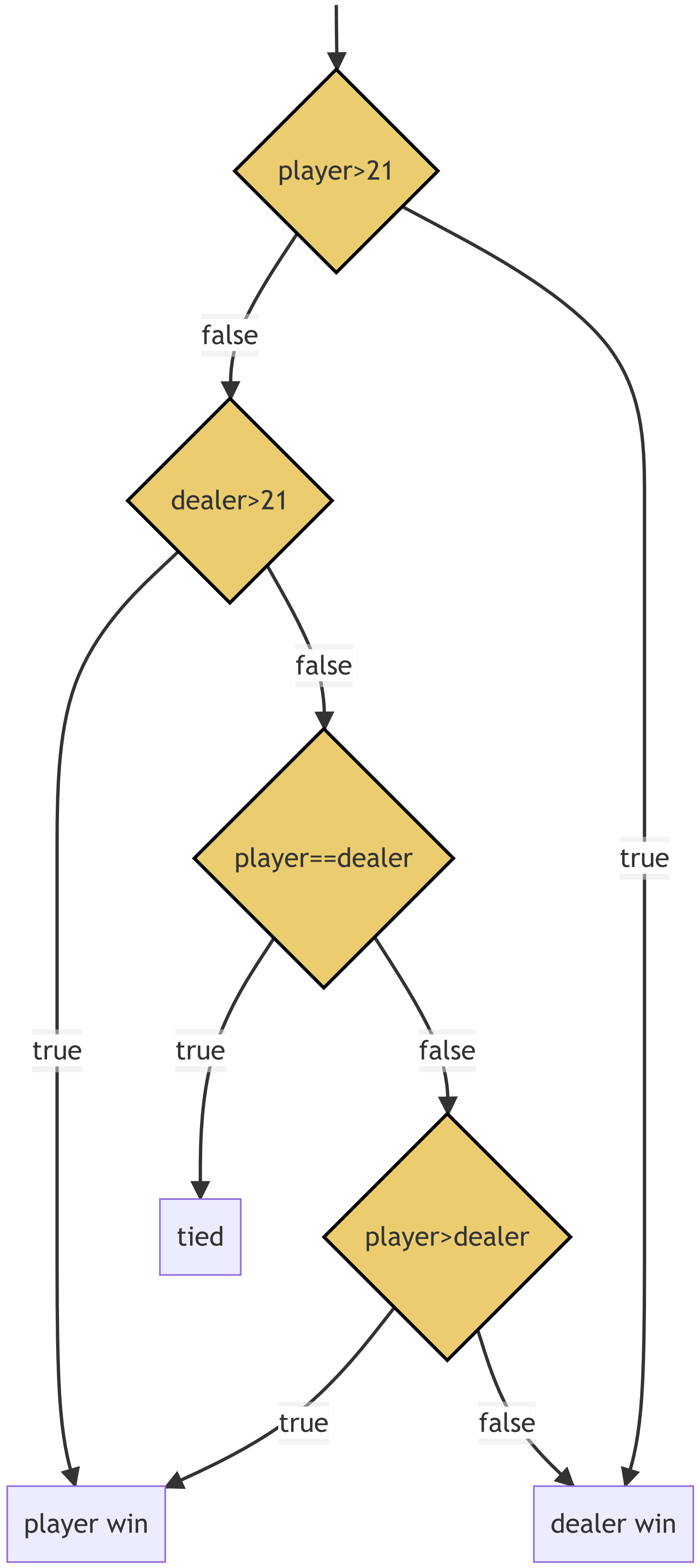

Dealer
deck:Deck
shuffle()
deal()
hit()
showHand()
dealer own the deck
Deck
topCardIndex:int
stackOfCards:BlackjackCard
shuffle()
getCard()
nextCard()
Card
face:str
suit:str
getValue()
stack of Cards
BlackjackCard
getValue()

Game logic



Code Optimization





- player get 4 Ace

- player get 3 Ace
- player get 2 Ace
- player get 1 Ace busted
- player get 1 Ace not busted
- Face.java
- Suit.java
- Card.java
- Deck.java
- Player.java
- Dealer.java
- Game.java

using logging

1. create a static logger instance of Logger class in Game class.
2. insert fine, info, severe message in our program.
3. While running the game, we setup logger lever to Level.WARNING for production, and setup Level.FINE for debugging.
4. insert log message into other classes by using logger defined in Game class.
5. send log message to a file so we can do analysis in the future.

❓How do I setup log level to SEVERE for Console?
✅😢the file handler also no output❌

```
logger.setLevel(Level.SEVERE);
```

❌😢it does NOT work for ConsoleHandler!
❌Set System property also faild
✅😄try logging.properties file
❓How can I make Console Handler and File Handler different log level?
✅use LogMager and provide logging.properties file.

1. create conf folder
2. create logging.properies file
3. add key-value pair
4. set System property fir the file

```
        static {
                System.setProperty("java.util.logging.config.file", "/Users/12818/workspace/java
        }
        public static Logger logger = Logger.getLogger("JOHN");
```

❓What is .properties file?
✔️It is a plain text file holds key-value pair separated by '=' for configuration.

```
handlers= java.util.logging.FileHandler, java.util.logging.ConsoleHandler
.level=FINEST

java.util.logging.FileHandler.level = FINE

java.util.logging.FileHandler.pattern = %h/workspace/java/mylogs%u.log
java.util.logging.FileHandler.limit = 50000
java.util.logging.FileHandler.count = 1
java.util.logging.FileHandler.maxLocks = 100
java.util.logging.FileHandler.formatter = java.util.logging.XMLFormatter

java.util.logging.ConsoleHandler.level = INFO
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
java.util.logging.SimpleFormatter.format=[%1$tF %1$tT] [%4$-7s] %5$s %n
```

Level setting Rule:

1. .level=FINE defined parent logger level for all handlers
2. java.util.logging.ConsoleHandler.level=WARNING defined log level for ConsoleHandler
3. log level overridden: which ever is more severe, logger will use that level

❓How do I send log message to file?
✔️

load logging.properties from file

```
static Logger logger = null;
static {
      System.setProperty("java.util.logging.config.file",
              "/Users/12818/workspace/java/huaxia/conf/logging.properties");
      //must initialize loggers after setting above property
      logger = Logger.getLogger("JOHN");
  }
```

😢👎This is not a good way, since if you deploy your application to different machine, the absolute folder path may not exists❗

load logging.properties from classpath

😄✔️Copy logging.properties into src folder is a good idea.👍

```
private static Logger logger;

  static {
      String path = LoggerExample4.class.getClassLoader()
                                   .getResource("logging.properties")
                                   .getFile();
      System.setProperty("java.util.logging.config.file", path);
      logger = Logger.getLogger("WANG");
  }
```

load logging.properties from InputSream

👌ok way to do logging configuration, but the code looks little complicated,
❓why choose it?

LoggerExample4.java

```
  static {
    InputStream stream = LoggerExample4.class
        .getClassLoader()
        .getResourceAsStream("logging.properties");
    try {
      LogManager manager = LogManager.getLogManager();
      manager.readConfiguration(stream);
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
```

❗⚡ the logging.properties file is put under ./src folder, and then copied to bin folder by compiler.

Understand log message format

1. java.util.logging.XMLFormatter (FileHandler default)
2. java.util.logging.SimpleFormatter

```
java.util.logging.SimpleFormatter.format=[%1$tF %1$tT] [%4$-7s] %5$s %n
```

```
[2021-08-04 14:17:02] [SEVERE ] this is a severe message
```

```
String.format(format, date, source, logger, level, message, thrown);
//position:             1     2       3       4       5       6
```

❓what is the simple example below?

```
java.util.logging.SimpleFormatter.format="%4$s: %5$s [%1$tc]%n"
```

✔️This prints 1 line with the log level $(4), thelogmessage(5)$ and the timestamp (1$) in a square bracket.

```
WARNING: warning message [Tue Mar 22 13:11:31 PDT 2011]
```

refer to java.util.Formatter class

where %s means String output, $4 means 4th item in the String.format which is level.$ means 1th item in the String.format which is date.

Sample logging.properties file
Use logging.properties

Integration Test

❓What is Integration Test?
✔️Play the Blackjack game by running Game class.

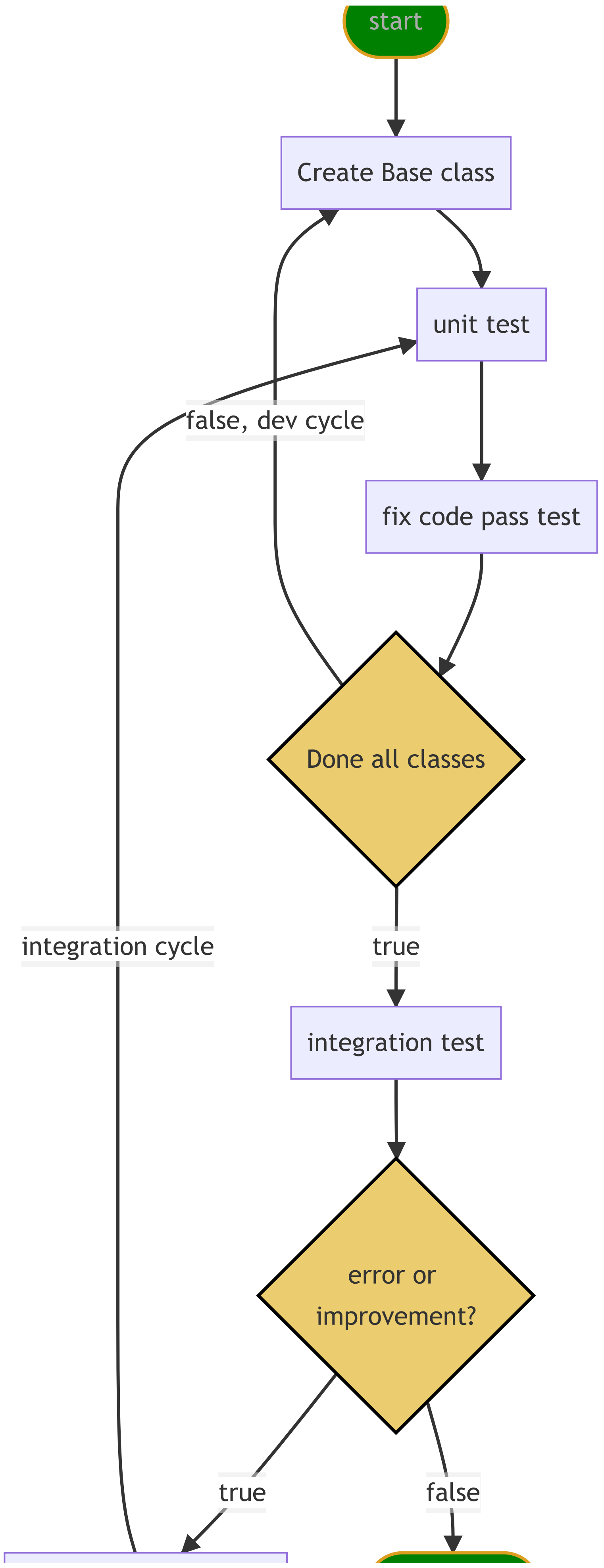
Documentation

Java Doc

```
cd blackjack/doc
mkdir api
cd api
javadoc -sourcepath ../../src -subpackages com.huaxia
```

Software development life cycle

- Test Driven Development (TDD)


start
Create Base class
unit test
false, dev cycle
fix code pass test
Done all classes
integration cycle
true
integration test
error or improvement?
true
false

Deployment(jar file)

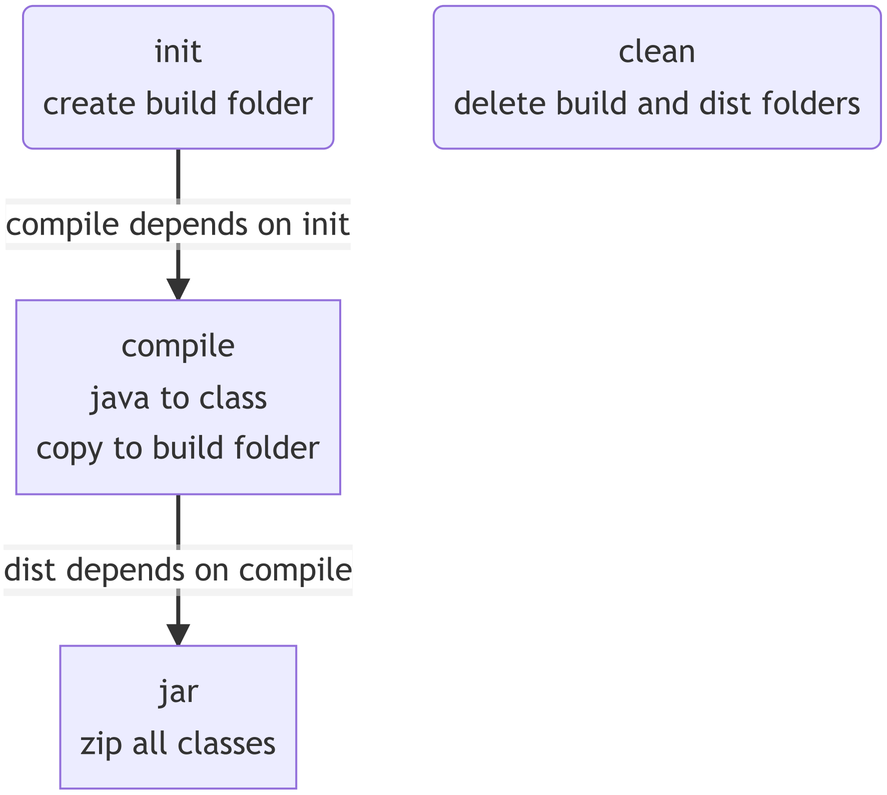
create jar file use ant

build.xml

```
<project name="blackjack" default="dist" basedir=".">
        <!-- set global properties for this build -->
        <property name="src" location="src" />
        <property name="build" location="build" />
        <property name="dist" location="dist" />

        <path id="project.classpath">
                <pathelement location="bin" />
        </path>

        <target name="init">
                <!-- Create the time stamp -->
                <tstamp />
                <!-- Create the build directory structure used by compile -->
                <mkdir dir="${build}" />
        </target>

        <target name="compile" depends="init" description="compile the source">
                <!-- Compile the Java code from ${src} into ${build} -->
                <javac srcdir="${src}" destdir="${build}"/>
        </target>

        <target name="dist" depends="compile" description="generate the distribution">
                <!-- Create the distribution directory -->
                <mkdir dir="${dist}/lib" />

                <!-- Put everything in ${build} into the snake.jar file -->
                <jar jarfile="${dist}/lib/blackjack.jar" basedir="${build}">
                        <manifest>
                                <attribute name="Built-By" value="John Wang" />
                                <attribute name="Main-Class" value="com.huaxia.blackjack.Game" /
                        </manifest>
                </jar>
        </target>
        <target name="run">
                <java fork="yes" classname="com.huaxia.blackjack.Game" failonerror="true">
                        <classpath refid="project.classpath" />
                </java>
        </target>

        <target name="clean" description="clean up">
                <!-- Delete the ${build} and ${dist} directory trees -->
                <delete dir="${build}" />
                <delete dir="${dist}" />
        </target>
</project>
```

- Targets

1. init
2. compile
3. dist
4. clean



use jar

```
cd blackjack/dist/lib
java -jar blackjack.jar
```

view jar

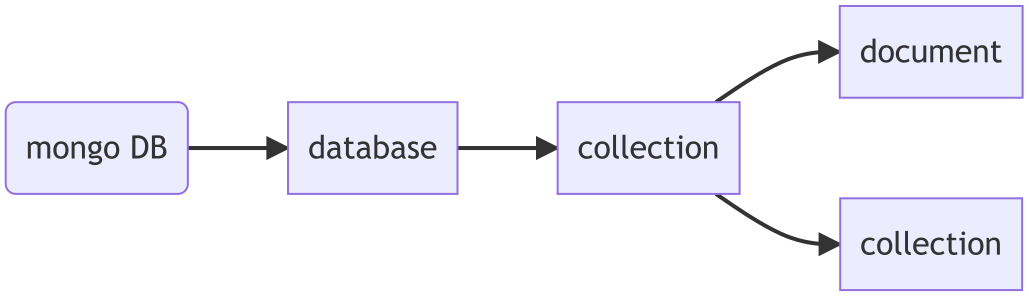
Help menu ⟹ Eclipse Marketplace... ⟹ Eclipse Archive Utility 0.1.0

Window ⟹ show view ⟹ Project Explore ⟹ click arrow on the jar file

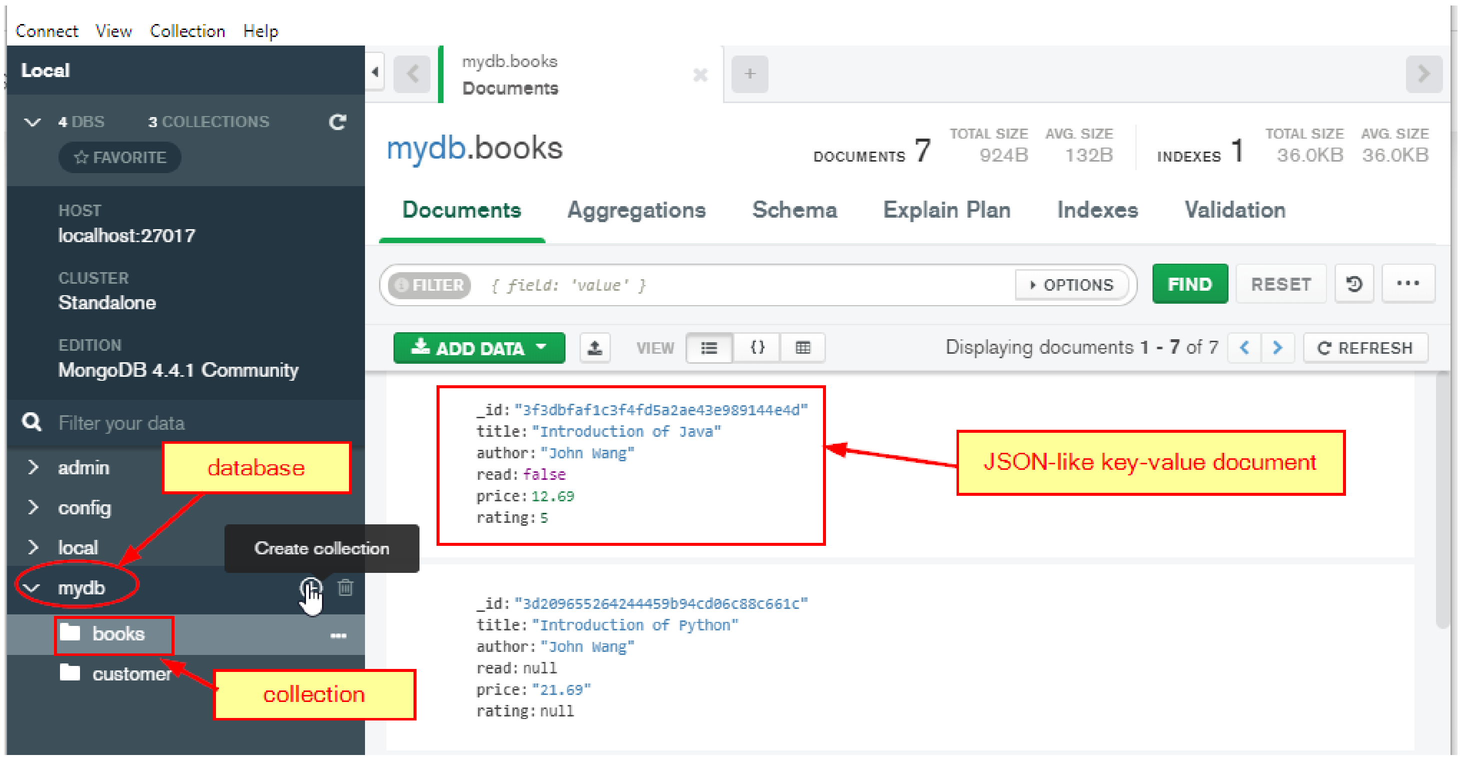
MongoDB

Using MongoDB in Java
❓What is MongoDB?
✅One of NoSQL database application written in C++.

1. stores data in JSON-like documents that can have various structures
2. uses dynamic schemas, which means that we can create records without predefining structure such as SQL relational database table.
3. the structure of a record can be changed simply by adding new fields or deleting existing ones.



4. document database
5. key-value database


Connect View Collection Help
Local
4 DBS 3 COLLECTIONS
FAVORITE
HOST
localhost:27017
CLUSTER
Standalone
EDITION
MongoDB 4.4.1 Community
Filter your data
admin
config
local
mydb
books
customer
Create collection
database
collection
mydb.books
Documents
DOCUMENTS 7 TOTAL SIZE 924B AVG. SIZE 132B
INDEXES 1 TOTAL SIZE 36.0KB AVG. SIZE 36.0KB
Documents Aggregations Schema Explain Plan Indexes Validation
FILTER { field: 'value' }
OPTIONS FIND RESET
ADD DATA VIEW
Displaying documents 1 - 7 of 7
REFRESH
JSON-like key-value document

```
_id: "3f3dbfaf1c3f4fd5a2ae43e989144e4d"
title: "Introduction of Java"
author: "John Wang"
read: false
price: 12.69
rating: 5
```

```
_id: "3d209655264244459b94cd06c88c661c"
title: "Introduction of Python"
author: "John Wang"
read: null
price: "21.69"
rating: null
```

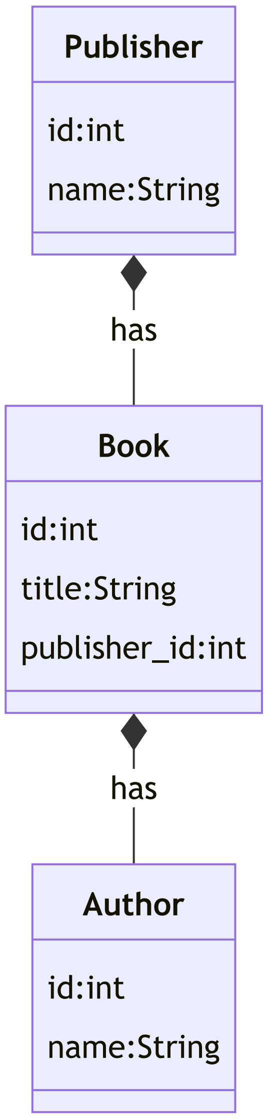
❓What is NoSQL database?
✔NoSQL databases (aka "not only SQL") are non tabular, and store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document, key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of data and high user loads.

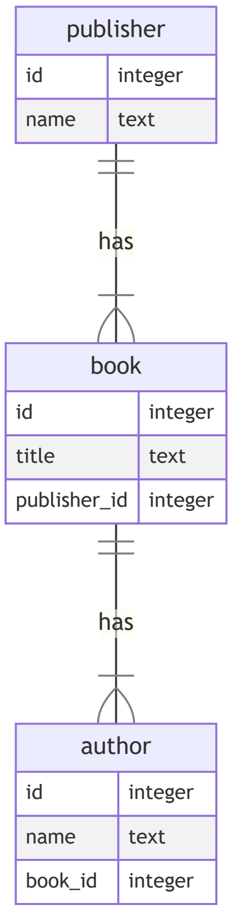
❓What is SQL?
✔SQL stands for Structured Query Language specially for relational database.
SQLite: Python built in SQL database.

- Java MongoDB API
- MongoDB Connection, Collection, CRUD
- Book.java

> In NoSQL, you don't design your database based on the relationships between data entities. You design your database based on the queries you will run against it. Use the same criteria you would use to denormalize a relational database: if it's more important for data to have cohesion (think of values in a comma-separated list instead of a normalized table), then do it that way.

Incremental Map/Reduce





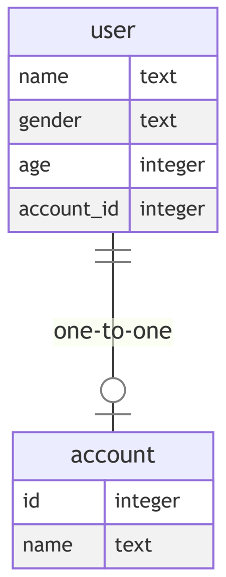
SQL==>Join vs. NoSQL==>Collation

```
SELECT
  publisher.id, publisher.name, book.title
FROM publisher
JOIN book ON publisher.id=book.publisher_id
ORDER BY publisher.id, book.title
```

```
{
  "_id":"oreilly",
  "collection":"publisher",
  "name":"O'Reilly Media",
  "books":[
    {"title":"CouchDB: The Definitive Guide"},
    {"title":"RESTful Web Services"},
    {"title":"DocBook: The definitive Guide"},
    {"title":"Building iPhone Apps with HTML, CSS, and JavaScript"}
  ]
}
```

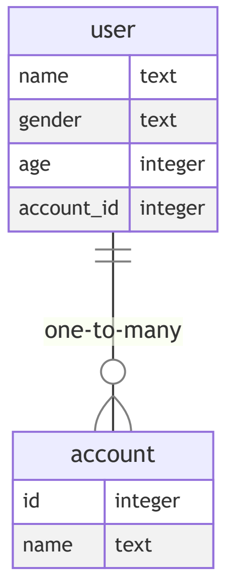
Entity Relationships in a Document Database

one-to-one relation

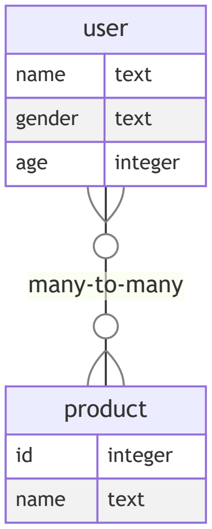


- when there is an exclusive need for getting the account data without the user data.

one-to-many relation



many-to-many relation



SQLite

❓ What is relational database (RDBMS)?
✅ A relational database is a type of database that stores and provides access to data points that are related to one another.

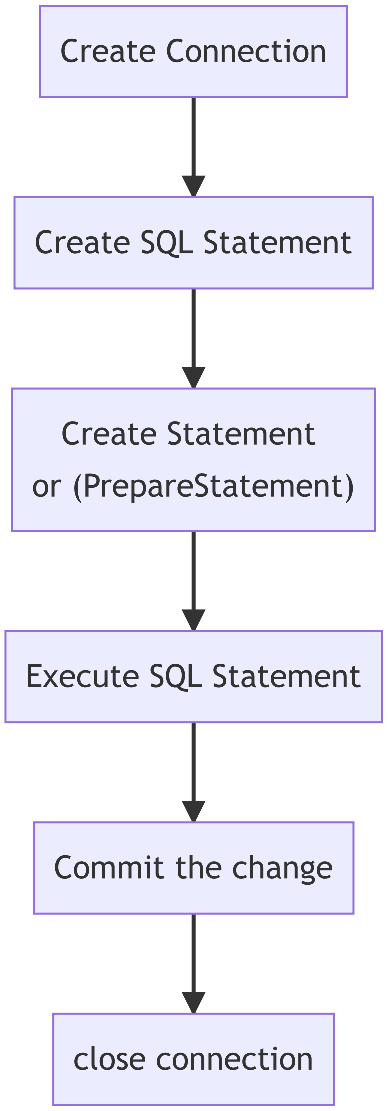
SQLiteDB Connection, CRUD

1. create()
2. retrieve()
3. update()
4. delete()

- Prepare for Relational database

1. create database (file)
2. create tables in the database

- to access the database



- Book.java
  - hide database complexity from Book user
- Test.java
- Hide db access complexity
- Test DB access function
- Create Syntax

```
INSERT INTO <table name> VALUES (?,?,...)
INSERT INTO <table name> (column1, column2, ...) VALUES (?, ?, ...)
```

- Retrieve Syntax

```
SELECT * FROM <table name>
SELECT * FROM <table name> WHERE <condition>
```

- Update Syntax

```
UPDATE <table name> Set column1=?, column2=?, ... WHERE <condition>
```

- Delete Syntax

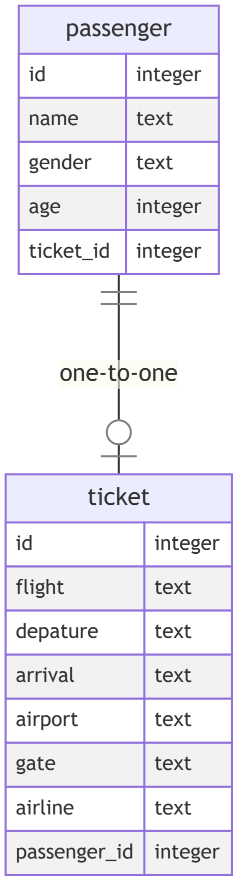
```
DELETE FROM <table name> WHERE <condition>
```

- Create [Book.writeBookToDB()]
- Retrieve All Book.getAll()
- Retrieve Book.loadBookFromDB()
- Update Book.updateBookInDB()
- Delete [Book.deleteBookInDB()]
- Book can CRUD itself to DB.
- Use DBHelper to reduce duplicated code
- Use DBSetting to set default database file connection

Practice:

one-to-one relationship

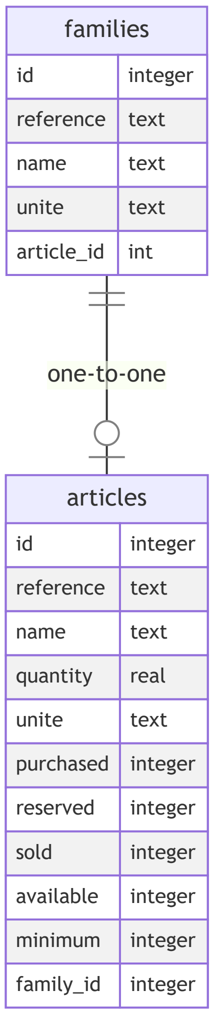
- Passenger vs. ticket



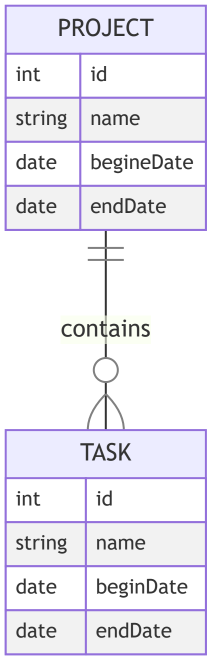
Passenger Database Access

Ticket Database Access

Ticket-Passenger Test

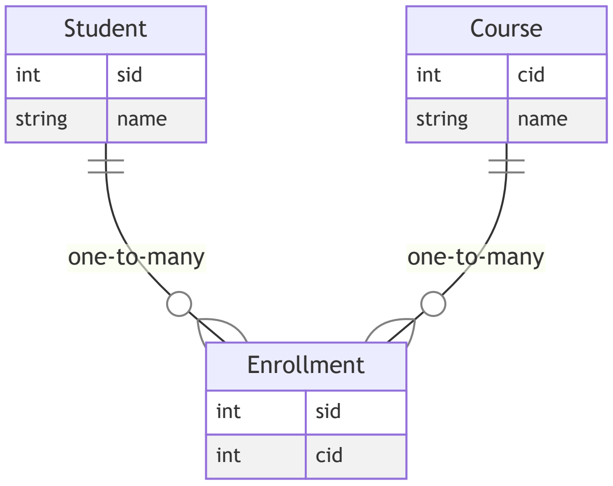


one-to-many relationship



- one to many
- Project > Task
- Test.java
- load project with all task
- get project from task
-

many-to-many relationship

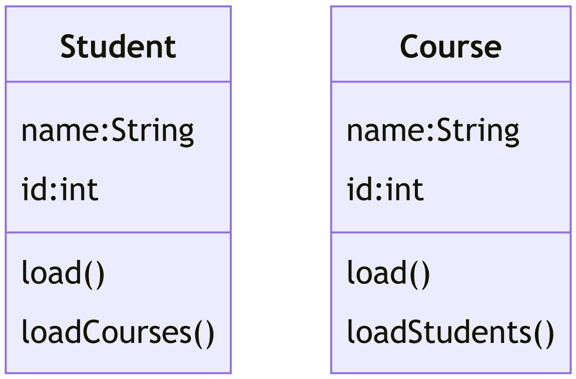


where the intermediate table **Enrollment** is called linking or conjunction table.

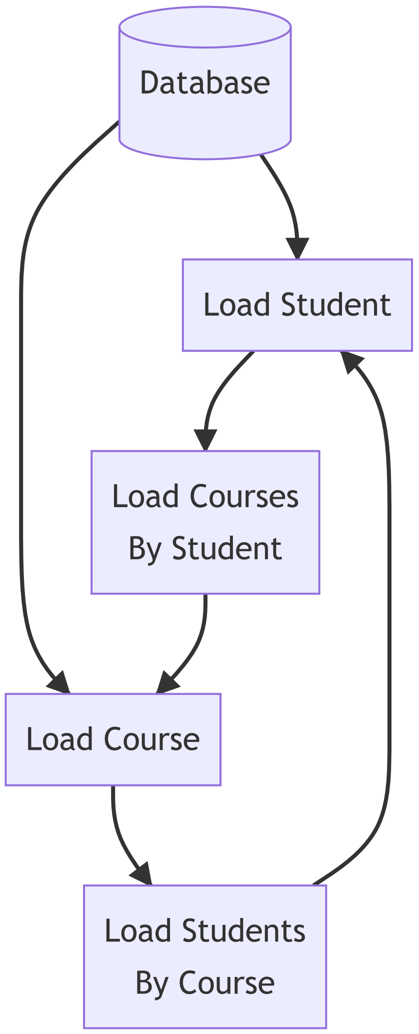
SQL = SELECT Student.name FROM Student JOIN Enrollment On(Student.sid=Enrollment.sid) WHERE Course.name='CS4320' many to many, Course:Student

Student.java

Test.java



⚡❗🐛👎Infinite Circle causes application died.



👍 Solution #1:

✔💡 Load all courses while loading student by the same SQL statement.

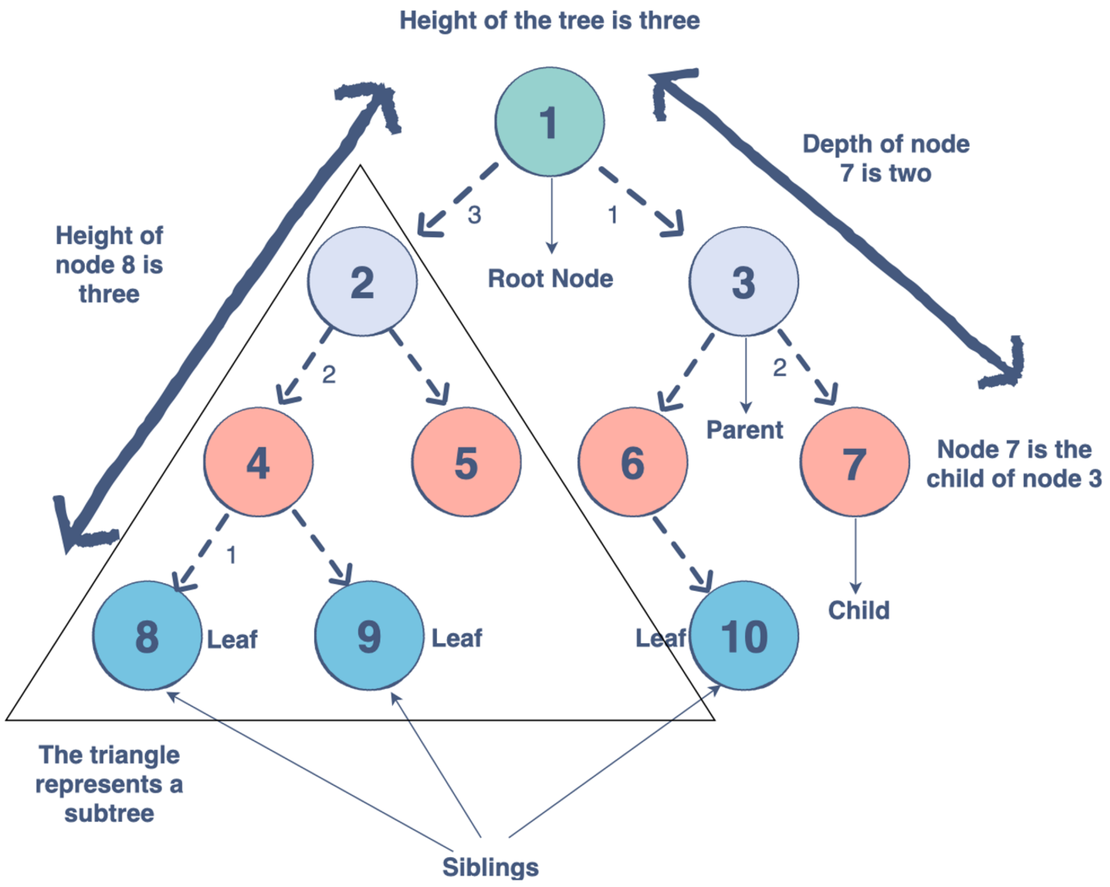
💡lazy loading: only load from DB when demands

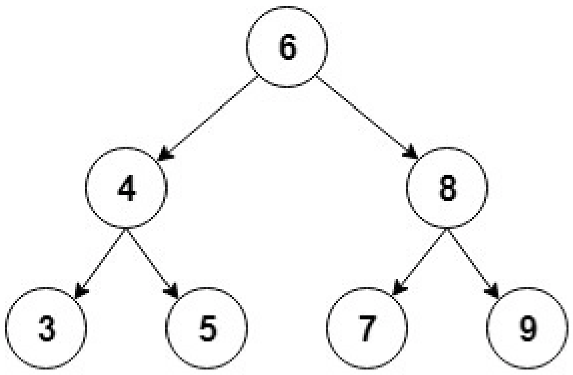
```
SELECT project.id, project.name, project.begin_date, project.end_date, task.id, task.name, task.
FROM project JOIN task on project.id=task.project_id
```

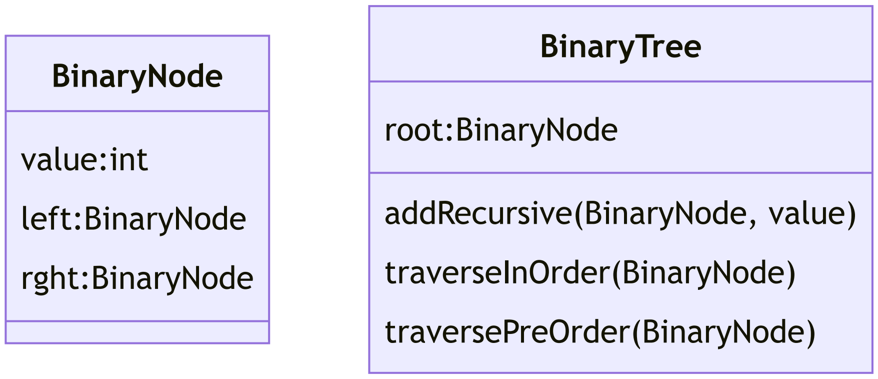
Student.loadCourses()
Course.loadStudents()
Test.manyToMany()

Data Structure

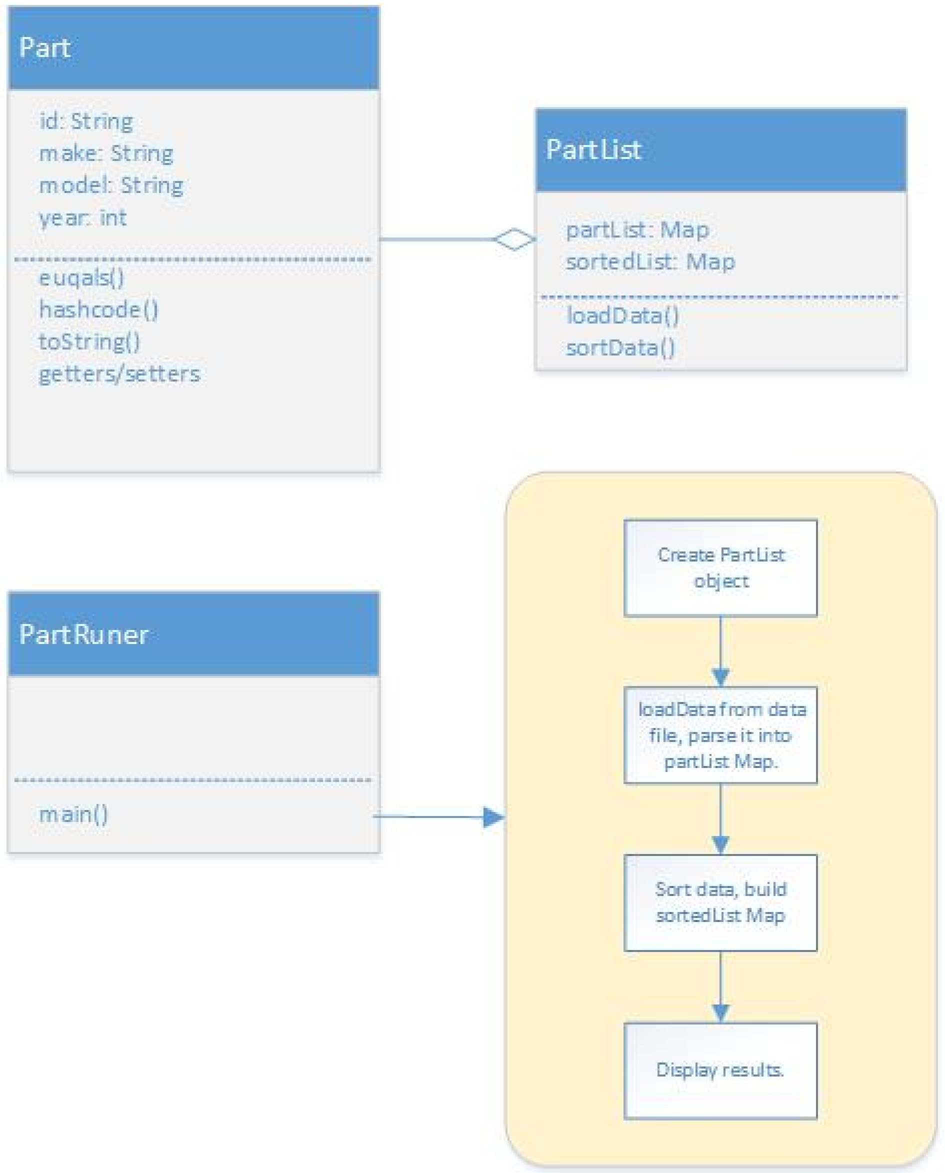
- Stack (FILO)
- Queue (FIFO)
- Linked List
- Doubly linked list
- Tree
  ❓ What is tree structure
  ✅Trees are a collection of nodes (vertices), and they are linked with edges (pointers), representing the hierarchical connections between the nodes. A node contains data of any type, but all the nodes must be of the same data type.


Height of the tree is three
1
Root Node
3
1
Depth of node 7 is two
Height of node 8 is three
2
3
2
2
4
5
6
Parent
7
Node 7 is the child of node 3
1
8
Leaf
9
Leaf
Leaf
10
Child
The triangle represents a subtree
Siblings

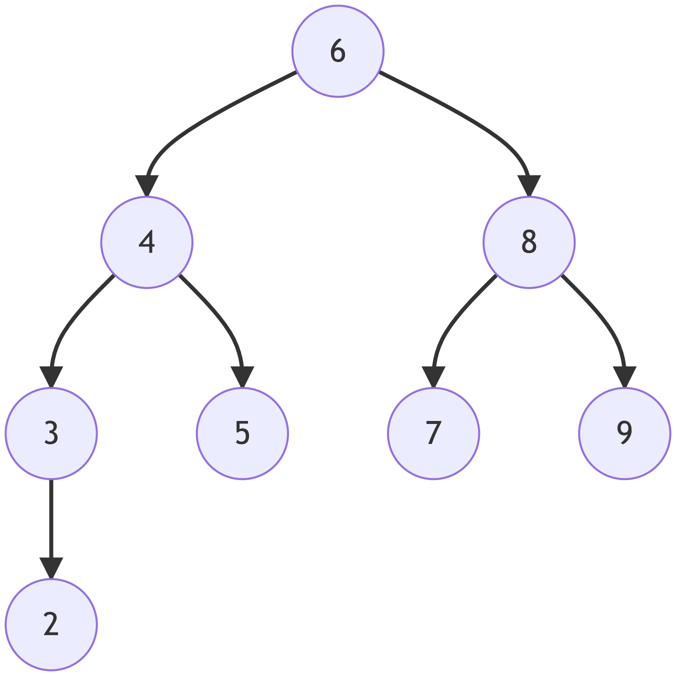



BinaryNode
value:int
left:BinaryNode
rght:BinaryNode
BinaryTree
root:BinaryNode
addRecursive(BinaryNode, value)
traverseInOrder(BinaryNode)
traversePreOrder(BinaryNode)

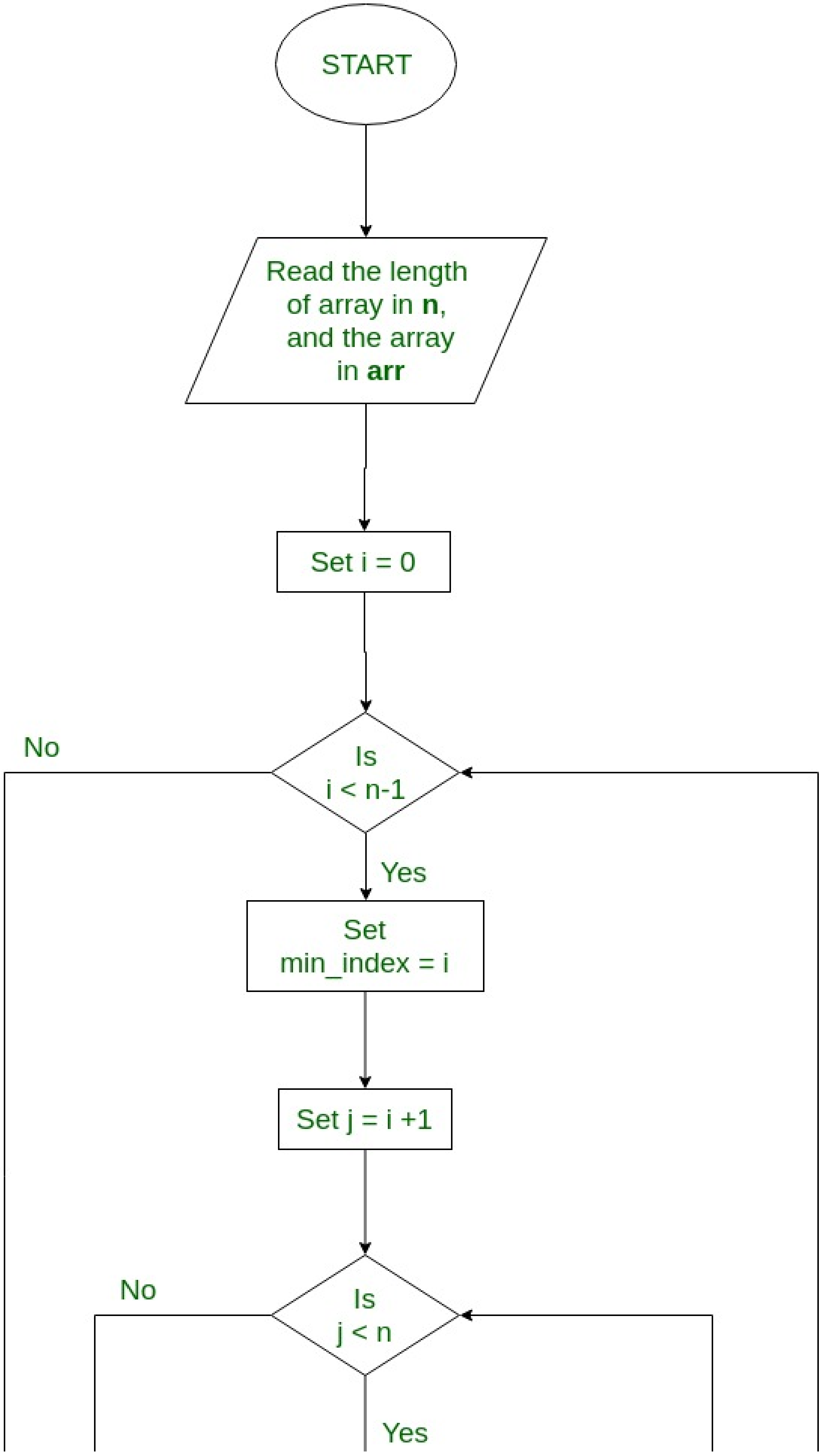
- BinaryNode
- BinaryTree
- PartList Homework
-

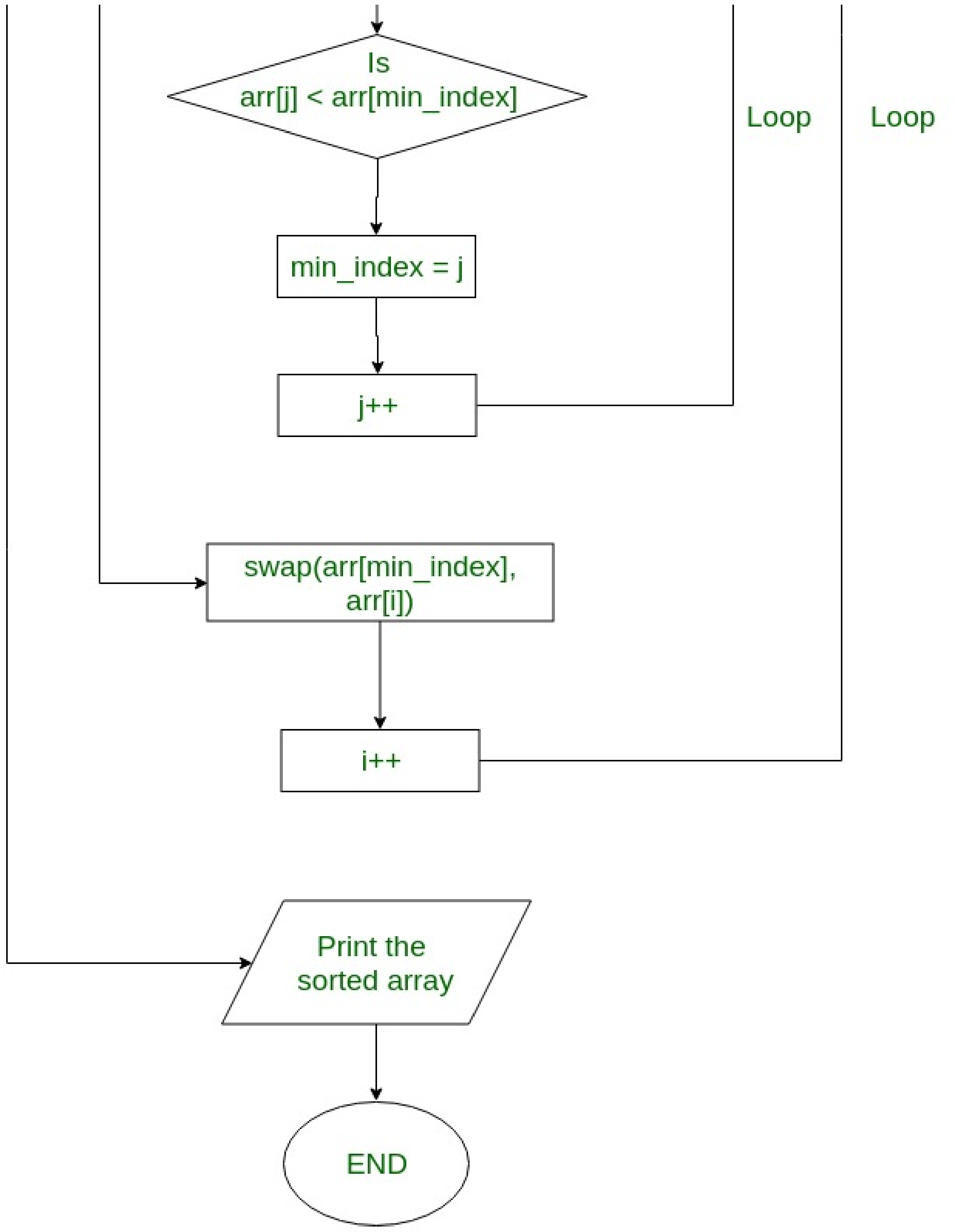


- BinaryTree.insert() test
- BinaryTree.remove() test
- BinaryTree



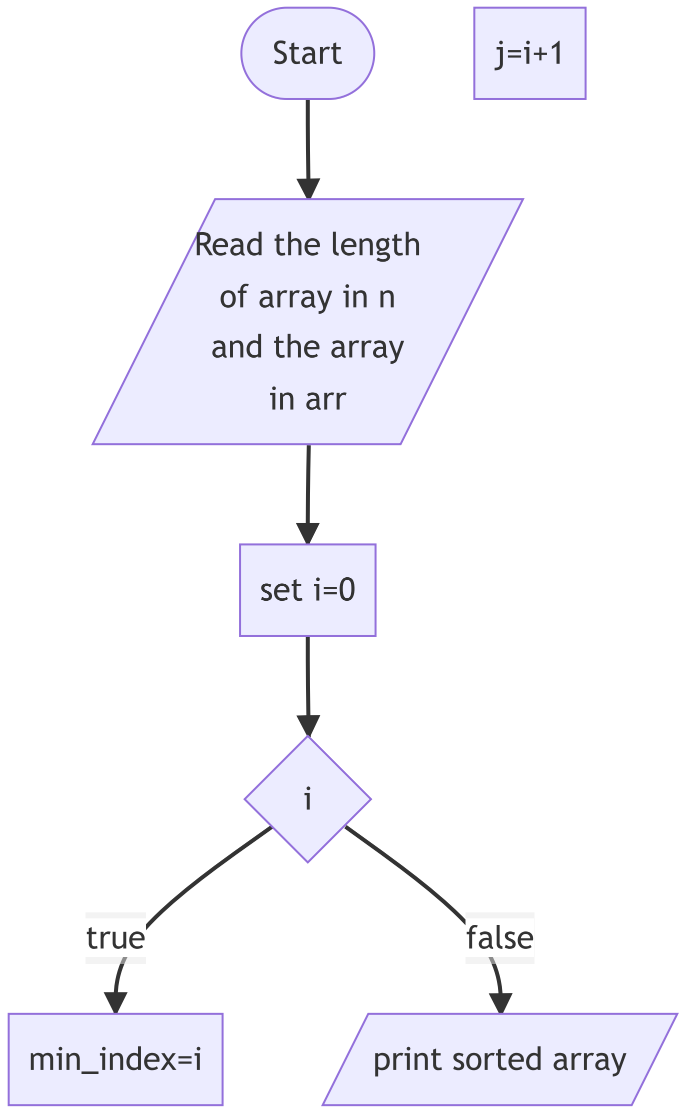
- TreeMap sort by AutoComparator
- Functional programing Better way to load part data
- Insertion sort

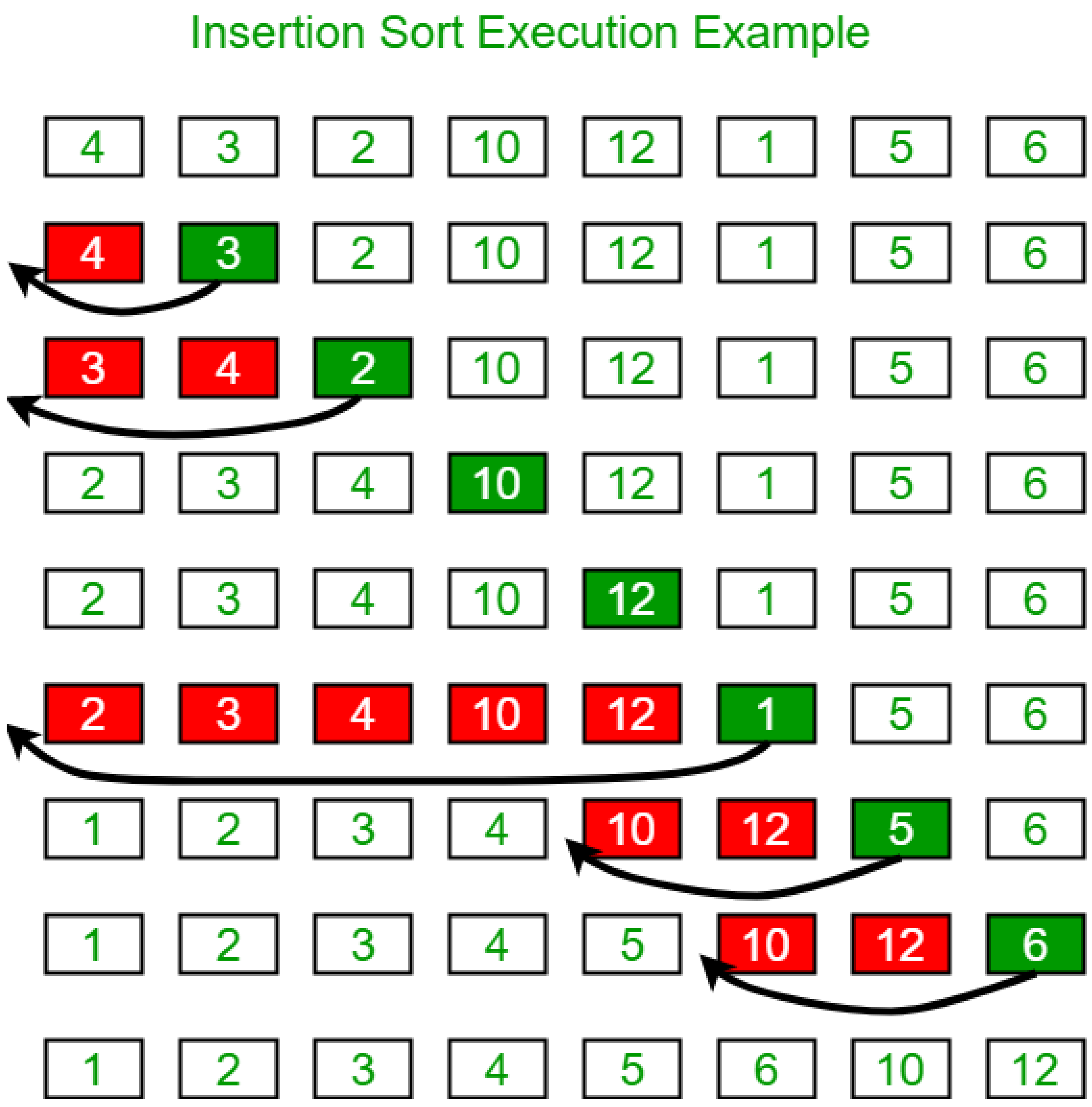

START
Read the length of array in n, and the array in arr
Set i = 0
No
Is i < n-1
Yes
Set min_index = i
Set j = i +1
No
Is j < n
Yes



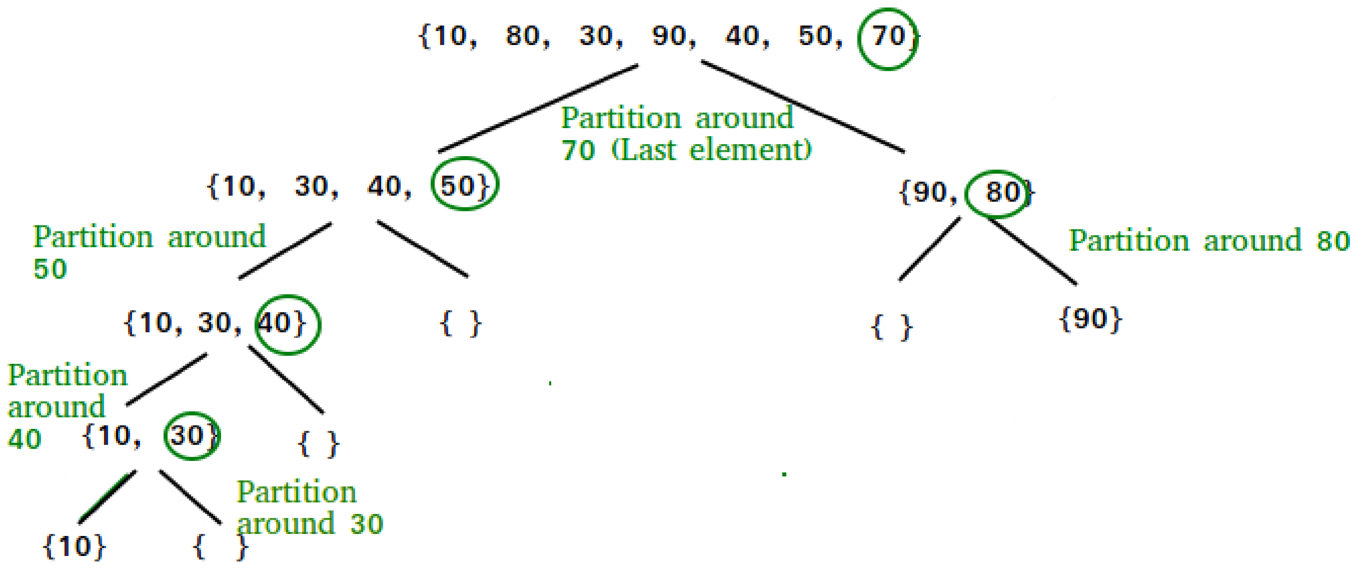
Flowchart for Selection Sort

- Selection Sort


Start
j=i+1
Read the length of array in n and the array in arr
set i=0
i
true
false
min_index=i
print sorted array



- Quick sort



- Order Part by make and year
- Sorted by make, mode, and year

- TreeMap Generics sorted by value
- TreeSet
- TreeSetDemo
- Video for insertion sort

Java Module

YouTube Understand Java Module

References

- 👍 All excercises
  1. Loop Excercises
  2. method Excercises
  3. class excercises
  4. OOP Excercises
  5. Operator Excercise
- Java Tutorial
- Good Java Tutorial WebSite
- Java Point
- Linked List