7. Java IO

- Java IO
- Stream
- File
- Exception
- Types of Exception
- Exception handling
- Custom Exception

7.1 Java IO

- Java IO is used to process the input and produce the output.

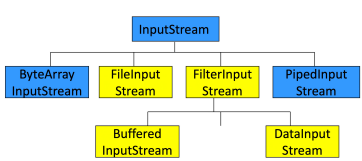
7.2 Streams

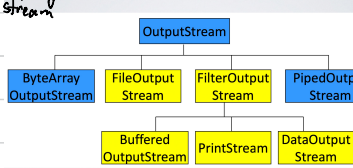
{ input stream
{ output stream

Java: 3 type of stream are created automatically { System.out, System.in, System.err

Types of Stream:
{ Byte Stream : handling input & output of byte. { InputStream, OutputStream (Image / mp3)
{ Character Stream : Handling character { Reader, Writer (txt file)

Anything dealt with character, use char stream





throw IO Exception

Most methods in this class return void and throws IO Exception in case of I/O errors

7.3 File

- The file reader class creates a reader that you can use to read the content of the file.
- { FileReader : create a reader that you can use it to read content of data
  FileWriter

7.4. Other Type of stream

Buffered Stream : each request is handled directly by OS, improve memory

Data Stream : used to support primitive data type value.

7.5 File cont.

- File defined by java.io but it's not for reading / writing
  It's just show properties of a file.

7.6 Exception very important

- Unwanted / Unexpected event, that disrupt normal program flow
  eg : ArithmeticExpection
  ArrayIndexOutOfBound
  IOException
  FileNotFoundException

- Error vs Exception
  Error : a serious logic problem that cannot be recovered. And most error is thrown by JVM
  Exception : indicates conditions that a reasonable application might try to catch

- Types of Exception
  - Checked : by the compiler and compiler time
    - eg: IO Exception, SQL Exception
  - Unchecked : Checked by JVM, at run time
    - Array out of Bound, Null PointerException

- Exception Handling
  - try
  - catch ①
  - throws
  - finally

① Try catch

```
static void method2()
{
        System.out.println("IN Method 2, Calling Method 3");
        try{
                method3(); }
        catch(ArithmeticException ae)
        {
                System.out.println ("Arithmetic Exception Handled: " +ae);
        }
        catch(Exception e)
        {
                System.out.println("Exception Handled");
        }
System.out.println("Returned from method 3");
}
```

Order: ∵ Arithmetic is child of Exception, if change order → compiler error

↑ Must before it's parent, otherwise (compiler error)

- order is important
- Can use multicatch like catch(Exception1 | Exception2 | Exception3)
- Can be placed within any method

② THROW

- added method signature to let other method know about what exception the called method can throw.
- Caller method can decide whether they want to handle or just pass the exception

③ Finally:

- execute in all circumstance
  - if the exception occur
  - in normal cases
- Mandatory statement can put in finally

```
class Test
{
    String str = "a";

    void A()
    {
        try
        {
            str +="b";     // abdef
            B();           // d
        }
        catch (Exception e)
        {
            str += "c";    // abc
        }
    }

    void B() throws Exception
    {
        try
        {
            str += "d";
            C();
        }
        catch(Exception e)
        {
            throw new Exception();
        }
        finally
        {
            str += "e";
        }

        str += "f";        // → No f ∵ Throw an exception
    }

    void C() throws Exception
    {
        throw new Exception();
    }

    void display()
    {
        System.out.println(str);
    }

    public static void main(String[] args)
    {
        Test object = new Test();
        object.A();
        object.display();
    }
```

- Custom Exception
  - Subclass of RuntimeException & Exception
  - Extend Exception: if want to create checked excep.
    - ~ RuntimeException unchecked ex.

why two types of exception?

We have 2 Types of Exception:

RuntimeException : unchecked, (Null pointer/Arithmetic/ArrayoutofBound) they are checked at runtime. (Runtime: unexpected behav/Program crashes)

Exception : checked (IO/SQL Exception). checked at compile time (Compile: usually syntax/reference).

(eg: Null)

[checked : must know about the exception so they can handle properly.

unchecked : may happend everywhere. Adding these exception to method declaration will/might reduce the program clarity, readability.

- Custom Exception: It's very useful when we want to handle specific exceptions related to our business logic

Day 8 Thread

Outline

- Thread & process
- Thread in Java { Thread / Runnable
- Daemon Thread
- Synchronization
- Thread problem

8.1 Thread vs. process

process : has a self-contained execution environment & own memory space

just a running program and isolate each other

Thread : smallest unit of processing, seperate path of processing

Threads are independant

It shares common memory data. each threads can communicate each other by sharing memory

{ thread
  runnable

8.2 Thread in Java

new thread

- create new thread { Extends the Thread class
  Implement Runnable interface
- syntax: public class Thread extends Object implements Runnable
  the functionality of the thread can only be achieved by overriding this run() method
- run() vs. start()

Thread & Runnable

{ run: can run this again
  start: create new thread, cannot do twice. when a program calls on start, a new thread created, then run() executed.

- Runnable Interface
  - syntax: public interface runnable
  - The implementing class must also override a void method name as run()
  - example: public class HelloRunnable implements Runnable {
    public void run() {
    }
    }

- Summary: Thread vs. Runnable
  - Recommend using Runnable rather than thread
  - Creating implementation of Runnable and passed it to a new Thread class uses aggregation rather than inheritance, which gives more flexibility

```
public class HelloRunnable implements Runnable {

    public void run() {        can run thread again,
                               but cannot start the thread again.
        System.out.println("Hello from a thread!");
    }

    public static void main(String args[]) {
        (new Thread(new HelloRunnable())).start();
    }                                      ↓
                                   create a new thread

}
```

- Main method:
  - default thread in every Java program: main thread
  - when start execution, JVM will create main thread

- Thread Life cycle:
  - Five State of a thread:
    - new : just create but not start
    - runnable : create, started and able to run
    - running : thread is currently running
    - blocked : thread is created, started and unable to run, because it's waiting other event occur
    - dead : thread is stopped
  - Blocked Thread
    - join : allowed thread wait for completion of another
    - sleep: causes current thread stop execution for a period time
    - both of them handle InterruptedException

8.3 Deamon Thread

-. Intro: - low priority thread
- Example: garbage collection
- when all user thread dies, JVM terminate the thread automatically
- Create deamon thread: public void setDaemon(boolean status)

8.4 Synchronization

-. Why Synchronization?

Threads communicate by sharing memory/field. but that can lead 2 type of problem; the communication is efficient. but,

- Thread Interference
- memory consistency error

=> Solve: Synchronization

=. Achieve synchronization:

- By using synchronized keyword with method definition.
- . . . . . with any block of code.

Thread Interference: A situation where two thread operating on same object at same time. Dealing with two thread.

Memory consistency: is about visibility and deal with hardware memory.

Synchronized method: when one thread that invoke a synchronized object, all other thread that invoked synchronized methods for same object blocked.

8.5 Thread problem:

- Deadlock (unable to gain access to shared resource and unable to make progress)
- Starvation
- Live Lock (both threads are too busy to response each other)

(Optional)

-. Deadlock: - describes two or more thread are blocked, however, waiting for each other.

Day9 JDB

Outline:

- Introduction to JDBC
- Database handling using JDBC
  - Load the Driver
  - Establish a connection
  - SQL Statement
  - Execution
- Transaction Management JDBC
- Exception Handling in JDBC
- Batch Processing in JDBC
- Advanced JDBC

9.1 Introduction

-. JDBC stands for "Java Database Connectivity"

9.2 DB handling using JDBC

-. JDBC use few steps to connect & do database insert/delete...

- Load the driver
- Establish Connection
- Create Statement
- Execute query and obtain result
- Iterate through result.
- Close connection

9.1 Load the Driver:

JDBC Driver is a software application that allowed java application interact with db

4 type of JDBC driver:

JDBC-ODBC Bridge Driver
- Adv: easy to use & Connect
- Dis: Performance degraded
  ODBC driver needs to installed on client machine

Native API driver (partially Java driver)
- use client side library
- Adv: Performance upgraded than JDBC-ODBC driver
- Dis: Native Driver & Vendor client library need to install

* most common use

Network protocol driver (full JD)
- Adv: no client side library is required
- Dis: requires network connection, on client side maintenance, maintenance of network connection is required and costly

Thin driver (full)
- converts JDBC directly to vendor database protocol.
- Adv: Better performance than any other driver. No software is required in client side
- Dis: Driver depend on database

op co

- Load the driver
- Establish Connection
- Create Statement
- Execute query and obtain result
- Iterate through result.
- Close connection

2.2 Establish Connection

- DriverManager class
- If any problem occurs during accessing database, it will throw a SQLException
- Syntax: Connection con = DriverManager.getConnection(databaseURL);

2.3 Create Statement

- Statement
- PreparedStatement : for values that inserted to DB again and again.
- CallableStatement : call stored procedure

Syntax: Statement stmt = con.createStatement();
PreparedStatement stmt = con.prepareStatement(String query);
CallableStatement stmt = con.prepareCall("{");

2.4 Execute Query

- executeQuery() : return a single result set — ResultSet rs = stmt.executeQuery("select * from emp");
- executeUpdate(): used for DDL and DML statement. like insert, update delete...

both of them throws SQL Exception.

- Iterate result set:

```
while (rs.next()){
    System.out.println(rs.getString(1));
    ...
}
```

2.5 Database Metadata

- Basically means the data that provide information of other data.
- Use of database Metadata API:
  DB users, tables, stored procedure ect.
  Information about primary key, foreign key

2.6 Close Statement:

- Just like file I/O, we have to close the db after we finished all jobs
- Syntax:
  Statement.close();
  connection.close();

9.3. Transaction management in JDBC

- What: Transaction represents a single unit of work
- ACID property (describes transaction management well)

  Atomicity: means either successful or null

  Consistency: means database from one consistency state to another consistency state

  Isolation: makesure every transaction is isolated from another.

  Durability: once transaction has been committed, it will remain so. even if there has error or power loss.

3 method of transaction Management in JDBC:
- commit() : submit transaction
- rollback() : cancel transaction
- setAutoCommit() : each transaction commited by default

9.4 Exception Handling

Program should recover or leave database in consistent state.

How finally help: you can rollback() your transaction in catch, and close database in finally statement.

9.5 Batch processing in JDBC

- why, How
  - Instead of execute a single query, we can execute a batch (group) of query. It makes the performance faster
  - java.sql.Statement & java.sql.PreparedStatement support batch processing
  - Advantage of Batching: Fast performance

```
Statement stmt = conn.createStatement();
stmt.addBatch("insert into user420 values('90','abb',600)");
stmt.executeBatch();
```

9.6. Advanced JDBC

- Java Callable Statement interface
  - Callable statement is used to call the stored procedure.
  - Business logic on database is best use of stored procedure to improve performance, because those are precompiled.

- Scrollable result set
  - a scrollable result set is used for retrieve the data forward or backward but not updating data

    rs.relative(-3) // go backward 5 step.

    rs.relative(7) // go forward 7 step

- Updatable Result Set.

  A update result set allowed as update like Lete a column value or inserting data.

Day10 WebApplication

Outline:

- Web Application { HTTP Request / HTTP Response
- Client
- Web Server

10.1 Web Application

1-1 Web Application (what)

- A web application is a computer program that utilize web browser and web technology to perform tasks over the Internet.

- A web Application use combination of server-side script (like Java) and client side script (Javascript, HTML) to represent information to users

1-2. HTTP & HTTP Request

HTTP
- HTTP stands for Hyper Text Transfer Protocol. And it's used to structure request and response over the Internet
- HTTP transfer data from one point to another

HTTP Request:

- (what): HTTP request is a packet of information that one computer sends another computer to communicate something
- A HTTP Request contains following parts:
  - Request Line specify method token [like get, put....]
  - Headers (0 / more)
  - Optional Body

(cont.)

- Get: 最常用的方法. 用于请求服务器发送某个资源, 应该是安全和幂等的

  对于多个URL请求, 应该返回同样的结果

  The method requests of a specific resource.

- post: 向服务器提交数据, 比如完成表单数据的提交, 将数据提交给服务器处理

  Post method is used to submit an entity to a specific resource.

  (Causing change / has side effect for server side).

- PUT: 让服务器用服务器主体部分来创建一个由所请求的URL命名的文档.

  如果该文档存在的话, 就用新下文档代替它

  The put method replace all current representations of target resource with the request payload.

- DELET: 请求服务器删除指定URL所定义的资源

  Delete method delete specify resource.

post vs. put

post: used to create/add

let server decide URI, use post

initialy create the object

put: used to create/add and update

user can define their URL

update the object.

URL vs URI

- URL: uniform resource locator
  - URL is just locator
- URI: uniform resource Identifier
  - URI can be name, locator or both for an online resource

1-3 HTTP Response.

- HTTP response is the packet of information send by server to client
- HTTP response should include Information requested by client.
- HTTP Response also have same structure

Status Line

- syntax: Status Line: HTTP/1.1 200 OK
- status Code
  - 1xx: Informational
  - 2xx: Successful
  - 3xx: Redirection
  - 4xx: client error
  - 5xx: server error
- Reason Phrase

Header

- Response header, just like request header, contain ≥0 header lines
  But very uncommon to have zero Header line
- After status Line and Before Response Body

Optional Body of the request

- Response Body
- Contains source data that request by client

10.2 Client

- What: A client in web application usually refer to a server that present server interface.

- Language supported by Web browser.

  HTML
  - Stands for Hyper Text Markup Language
  - Describe the structure of the web page.
  - Tell the browser how to display the content

  CSS
  - stands for Cascating style sheet
  - describes how HTML element displayed.
  - It can save lots of work, it can control the layout of multiple pages all at once.

  JavaScript.
  - Is the programming language of HTML and Web
  - Program behavior of the webpage.
  - HTML and CSS will display static web page, JS will make it dynamically

10.3 Web Server

what is Web Server?

- A program that create webpage for user in response to their request.

4 Leading Web Server
- Apache HTTP server (like Tomcat)
- Internet Information Service
- Nginx
- Google Web Server

Terminology:
- SMTP
- FTP
- ISP
- DNS