* Exception Handling
* Multithreading
* Collections
* Java 8
* Spring Core
* Spring MVC
* Spring Boot
* Spring Microservices
* Angular

Pre-requisites

* create git-hub account
* install git from : <https://git-scm.com/downloads>
* Eclipse IDE
* Java 8

Why Java?

Java is a platform independent & Object oriented language, using which you can develop any kind applications like standalone, distributed, mobile applications, you can write programs to IC’s, Cards and etc

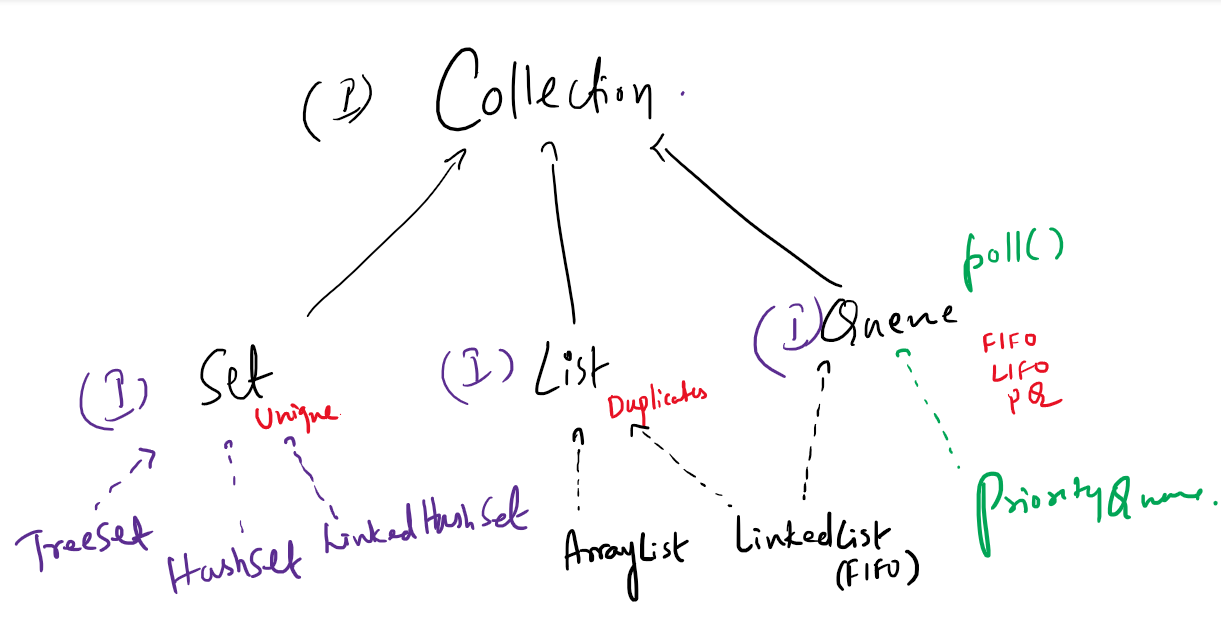
OOPs concepts:

1. Encapsulation: private variables & public setters & getters
2. Inheritance: extends keyword
3. Polymorphism: overriding and overloading
4. Abstraction: abstract class & interface

Exception Handling:

try, catch, finally, throw & throws

Collection Framework:



Collection has methods:

add, remove, clear, size, isEmpty, iterator,...

Set: Allows only unique elements

TreeSet: Maintains the elements in sorted order

HashSet: Maintains the elements in random order, but retrieval is faster

LinkedHashSet: Maintains the elements in insertion order

List: Allows duplicates and elements have index

ArrayList: Maintains elements in contiguous memory address, retrieval is faster but adding/removing of the elements are little slower

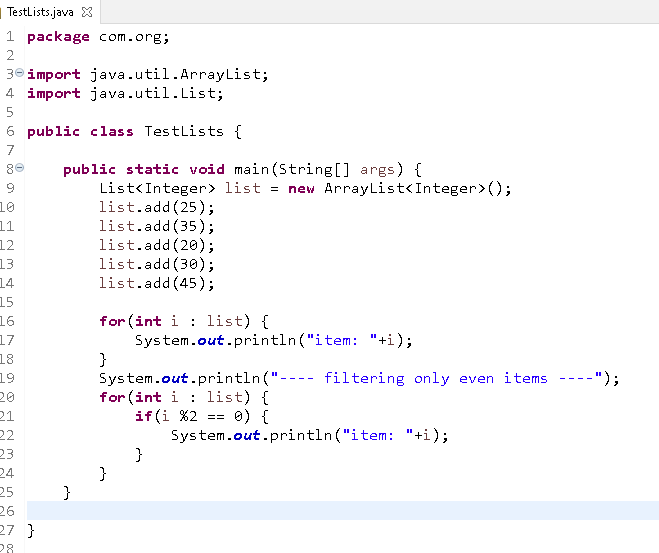
LinkedList: Maintains elements in non-contiguous memory address, retrieval is little slower but adding/removing of the elements are faster

Some useful git commands:

git clone <<url>>  
git pull  
git push -u origin master  
git add .  
git commit -m ‘message’  
git status

Collection:

List<Integer> list = new ArrayList<Integer>();  
ArrayList<Integer> list = new ArrayList<Integer>();



Customer, Employee, Student and so on....

Exercise:

Create a user class which will have id, name and password, generate setters & getters

Add 5 user objects inside List<User> with different id, name & password

Iterate the List<User> such that it will only display the names whose length is more than 5

Object class: toString(), equals() & hashCode():

Set internally calls equals() & hashCode() of Object class, but you can override them in your userdefined class to identify the duplicates

**package** com.org;

**public** **class** User {

**private** **int** id;

**private** String name;

**private** String password;

**public** User(**int** id, String name, String password) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.password = password;

}

**public** User() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

@Override

**public** String toString() {

**return** "User [id=" + id + ", name=" + name + ", password=" + password + "]";

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + id;

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

User other = (User) obj;

**if** (id != other.id)

**return** **false**;

**return** **true**;

}

}

TestSet.java

**package** com.org;

**import** java.util.HashSet;

**import** java.util.Set;

**public** **class** TestSets {

**public** **static** **void** main(String[] args) {

Set<User> set = **new** HashSet<User>();

set.add(**new** User(100, "Alex", "alex123"));

set.add(**new** User(103, "Alex", "alex123"));

set.add(**new** User(100, "Alex", "alex123"));

set.add(**new** User(100, "Alex", "alex123"));

System.***out***.println("Size: "+set.size());

}

}

Output:

Size: 2

Map: It maintains elements in key value pairs, it uses put() method to store & get() method to retrieve, here key should be unique.

Map is an interface and it has 4 implementations

1. HashMap: Random order, it is a newer class and supports null, not thread safe
2. Hashtable: Random order, it is an older class and doesn’t support null, it is threadsafe
3. TreeMap: Sorted order
4. LinkedHashMap: Insertion order

Exercise:

Create a map that stores multiple values for a single key

|  |  |
| --- | --- |
| Key(String) | Value (List<Student> |
| CS | Student1, Student2, .. |
| EC | Student3, Student4,.... |

And display the list of students based on the key

i.e, EC -> Student3, Student4,...

CS -> Student1, Student2,...