Summary of the First Week

Lingyan Xie

Future vs CompletableFuture

Futures were introduced in Java 5 (2004). They're basically placeholders for a result of an operation that hasn't finished yet. Once the operation finishes, the Future will contain that result. For example, an operation can be a Runnable or Callable instance that is submitted to an ExecutorService. The submitter of the operation can use the Future object to check whether the operation isDone(), or wait for it to finish using the blocking get() method.

CompletableFutures were introduced in Java 8 (2014). They are in fact an evolution of regular Futures, inspired by Google's Listenable Futures, part of the Guava library. They are Futures that also allow you to string tasks together in a chain. You can use them to tell some worker thread to "go do some task X, and when you're done, go do this other thing using the result of X". Using CompletableFutures, you can do something with the result of the operation without actually blocking a thread to wait for the result.

In summary, Future transfers single value using synchronous interface. CompletableFuture transfers single value using both synchronous and asynchronous interfaces

(When you execute something synchronously, you wait for it to finish before moving on to another task. When you execute something asynchronously, you can move on to another task before it finishes.)

Day 1

**Maven, Git, Data Type, String vs StringBuilder vs StringBuffer, String Constant Pool**

**Maven**

Maven is an automation tool used for java projects. There are three types of repositories, local, central, and remote.

Its life cycle:

-validate

-compile

-test

-package

-jar/ war

-verify

-install

-deploy

Whenever we open a Maven Project, first step is to read the POM.XML file. Groupid contains the company domain.

**Git**

Git is a DevOps tool used for source code management. It is a free and open-source version control system

**DataType**

primitive type: byte, short, int, long, float, double, char, boolean

wrapper class: Byte, Short, Integer, Long, Float, Double, Character, Boolean

Autoboxing and unboxing: auto-conversion that the java compiler makes between the primitive types and wrapper class(object).

**String**

-immutable

thread safe

**StringBuilder**

-mutable

-not thread safe

**StringBuffer**

-mutable

-thread safe

**String Constant Pool**

**Diagram

Description automatically generated**

== compares the address, equals() contains the content.

Also, all primitive types have constant pool.

Day2

**equals()&hashcode, collections, Comparable vs Comparator, JVM, class loader, garabage collector**

**equals()&hashcode**

**Text

Description automatically generated**

12. Why do we override equals and hashCode methods together?

Failure to do so will result in a violation of the general contract for Object.hashCode(), which will prevent your class from functioning properly in conjunction with all hash-based collections, including HashMap, HashSet, and Hashtable.

**Collection**

**Diagram

Description automatically generated**

Note that hashset does not contain insertion order and treeset is sorted by comparator

For other collections, please check class note.

**How does Hashmap work internally?**

**Table

Description automatically generated**

HashMap vs HashTable vs ConcurrentHashMap

Hashmap is not thread safe whereas hashtable an concurrenthashmap are thread safe. Also, hashtable uses a shared lock whereas concurrentHashmap uses 16 thread which makes it more efficient to use.

**Comparator vs Comparable**

**Text

Description automatically generated**

**JVM**

JVM - Java Virtual Machine, loads, verifies and executes Java bytecode. It allows Java programs to run on any device or operating system (known as the "Write once, run anywhere" principle), and to manage and optimize program memory.

Diagram

Description automatically generated

**Class Loader**

**Diagram

Description automatically generated**

**Garage Collection**

Oracle’s HotSpot is by far the most common. It offers a robust and mature set of garbage collection options.

While HotSpot has multiple garbage collectors that are optimized for various use cases, all its garbage collectors follow the same basic process. In the first step, unreferenced objects are identified and marked as ready for garbage collection. In the second step, marked objects are deleted.

Day3

**Keyword, OOP, Exceptions**

**Keyword**

Please check the keyword homework to see the full list.

Text, letter

Description automatically generated

How do we make an immutable class?

1. Add final keyword before class
2. Make private final fields
3. No setter, only getter
4. Return a deep copy of the collections for getter, since collections are mutable.

**Static**

**Static stores in the method area(class level) whereas others store in the heap area(object level)**

**Also, static can be inner class, but not in top-level class**

**OOP**

Abstraction: abstract class, interface

Encapsulation: private fields, getter and setter

Inheritance: extends, implements

Polymorphism: override, overload

**Exception**

Checked Exception is a compile-time exception and unchecked exception is a runtime exception. They both implements throwable.

Day4

**Generics, IO Stream, Serialization & Deserialization, Java 8 new features**

**Generics**

* easier and less error prone
* enforce type correctness at compile time
* without causing any extra overhead to your application

Text

Description automatically generated

In Java Collections, we use T for type, E for element, K for key and V for value. ? is wildcard

IO Stream

A continuous flow of data. Common IO Stream:

Byte Stream

* InputSteam, OutputStream
* 1 byte = 8 bits

CharaterStream

* Reader, Writer
* 2 byte = 16 bits

**Serialization & Deserialization**

Please view code in class notes.

We use transient keyword in some fields to avoid serialization, therefore, protecting sensitive data.

**Java 8 new features**

* Lambda, (argument) -> {body}, it is functional programming and has less code
* Functional interface

**Predicate**

**public boolean test(T t)**

**Function**

**public R apply(T t)**

**Consumer**

**public void accept(T t)**

**Supplier**

**public R get()**

Text

Description automatically generated

Use it with lambda

Also, it contains one abstract method and any number of default methods. SAM interface.

* Optional
* Stream API

Week2 Day6

**Multi-threading, Thread-pool**

Multi-threading

## thread vs process

* process
  + independent memory space, heap, OS resources
* thread
  + shared memory space
  + private stack, program counter, register

Thread States

Diagram

Description automatically generated

## thread creation

* extends Thread
* implements Runnable
* implements Callable
* thread pool

runnable vs callable

* no return / has
* no exception / has
* run() / call()

In industry, completableFuture is used(high level)

**ThreadPool**

ThreadPoolExecutor

* corePoolSize
* maximumPoolSize
* KeepAliveTime
* Time unit
* work queue
* thread factory
* handler
  + abortPolicy
  + callerRunPolicy
  + discardPolicy
  + discardOldestPolicy

**Diagram

Description automatically generated**

For Lock,

* + synchronized
  + Lock interfece

For this section, please read the code in classnotes.