Agenda

- Exception Chaning
- clone()
- Strings
  - String
  - StringBuffer
  - StringBuilder
- ~~Date/LocalDate/Calender~~
- ~~Enum~~
- ~~Garbage Collector~~

Exception chaining

- Sometimes an exception is generated due to another exception.
- For example, database SQLException may be caused due to network problem SocketException.
- To represent this an exception can be chained/nested into another exception.
- If method's throws clause doesn't allow throwing exception of certain type, it can be nested into another (allowed) type and thrown.

```
public static void getEmployees() throws SQLException {
        // logic to get all the employees from database
        boolean connection = false;
        if (connection) {
            // fetch data
            boolean data = false;
            if (data)
                System.out.println(data);
            else
                throw new SQLException("Data not found");
        } else
            throw new SQLException("failed", new SocketException("Connection
rejected"));
    }
```

Clone method

- The clone() method is used to create a copy of an object in Java. - It's defined in the java.lang.Object class and is inherited by all classes in Java.
- It returns a shallow copy of the object on which it's called.

```
protected Object clone() throws CloneNotSupportedException
```

- This means that it creates a new object with the same field values as the original object, but the fields themselves are not cloned.

- If the fields are reference types, the new object will refer to the same objects as the original object.
- In order to use the clone() method, the class of the object being cloned must implement the Cloneable interface.
- This interface acts as a marker interface, indicating to the JVM that the class supports cloning.
- It's recommended to override the clone() method in the class being cloned to provide proper cloning behavior.
- The overridden method should call super.clone() to create the initial shallow copy, and then perform any necessary deep copying if required.
- The clone() method throws a CloneNotSupportedException if the class being cloned does not implement Cloneable, or if it's overridden to throw the exception explicitly.

Cloneable interface

- Enable creating copy/clone of the object.
- If a class is Cloneable, Object.clone() method creates a shallow copy of the object.
- If class is not Cloneable, Object.clone() throws CloneNotSupportedException.
- A class should implement Cloneable and override clone() to create a deep/shallow copy of the object.

Strings

- java.lang.Character is wrapper class that represents char.
- In Java, each char is 2 bytes because it follows unicode encoding.
- String is sequence of characters.
    - 1. java.lang.String: "Immutable" character sequence
    - 2. java.lang.StringBuffer: Mutable character sequence (Thread-safe)
    - 3. java.lang.StringBuilder: Mutable character sequence (Not Thread-safe)
- String helpers
    - 1. java.util.StringTokenizer: Helper class to split strings

String Class Object

- java.lang.String is class and strings in java are objects.
- String constants/literals are stored in string pool.
- String objects created using "new" operator are allocated on heap.
- In java, String is immutable. If try to modify, it creates a new String object on heap.

```
String name = "sunbeam"; // goes in string pool

String name2 = new String("Sunbeam"); // goes on heap
```

- Since strings are immutable, string constants are not allocated multiple times.
- String constants/literals are stored in string pool. Multiple references may refer the same object in the pool.
- String pool is also called as String literal pool or String constant pool.
- From Java 7, String pool is in the heap space (of JVM).
- The string literal objects are created during class loading.

StringBuffer and StringBuilder

- StringBuffer and StringBuilder are final classes declared in java.lang package.
- It is used create to mutable string instance.
- equals() and hashCode() method is not overridden inside it.
- Can create instances of these classes using new operator only. Objects are created on heap.
- StringBuffer implementation is thread safe while StringBuilder is not thread-safe.
- StringBuilder is introduced in Java 5.0 for better performance in single threaded applications.