Agenda

- String
- String Pool
- StringBuffer
- StringBuilder
- String Tokenizer
- Enum
- ~~clone()~~
- ~~Date/LocalDate/Calendar~~

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.

```java
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.

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.

String Tokenizer

- Used to break a string into multiple tokens - like split() method.
- Methods of java.util.StringTokenizer
  - boolean hasMoreTokens()
  - String nextToken()
  - String nextToken(String delim)

Enum

- In C enums were internally integers
- In java, It is a keyword added in java 5 and enums are object in java.
- used to make constants for code readability
- mostly used for switch cases
- In java, enums cannot be declared locally (within a method).
- The declared enum is converted into enum class.
- The enum type declared is implicitly inherited from java.lang.Enum class. So it cannot be extended from another class, but enum may implement interfaces.
- The enum constants declared in enum are public static final fields of generated class.
- Enum objects cannot be created explicitly (as generated constructor is private).
- The enums constants can be used in switch-case and can also be compared using == operator.
- The enum may have fields and methods.

```java
public abstract class Enum<E> implements java.lang.Comparable<E>,
java.io.Serializable {
    private final String name;
    private final int ordinal;

    protected Enum(String,int); // sole constructor - can be called from user-
defined enum class only

    public final String name(); // name of enum const
    public final int ordinal(); // position of enum const (0-based)
    public String toString(); // returns name of const
    public final int compareTo(E);// compares with another enum of same type on
basis of ordinal number
    public static <T> T valueOf(Class<T>, String);
    // ...
}
```

```java
// user-defined enum
enum ArithmeticOperations {
    ADDITION, SUBTRACTION, MULIPLICATION, DIVISION
}

// generated enum code
final class ArithmeticOperations extends Enum {
```

```java
    private ArithmeticOperations(String name, int ordinal) {
        super(name, ordinal); // invoke sole constructor Enum(String,int);
    }

    public static ArithmeticOperations[] values() {
        return (ArithmeticOperations[])$VALUES.clone();
    }

    public static ArithmeticOperations valueOf(String s) {
        return (ArithmeticOperations)Enum.valueOf(ArithmeticOperations,s);
    }


      public static final ArithmeticOperations ADDITION;
      public static final ArithmeticOperations SUBTRACTION;
      public static final ArithmeticOperations MULIPLICATION;
      public static final ArithmeticOperations DIVISION;
      private static final ArithmeticOperations $VALUES[];

      static {
        ADDITION = new ArithmeticOperations("ADDITION", 0);
        SUBTRACTION = new ArithmeticOperations("SUBTRACTION", 1);
        MULIPLICATION = new ArithmeticOperations("MULIPLICATION", 2);
        DIVISION = new ArithmeticOperations("DIVISION", 3);
        $VALUES = (new ArithmeticOperations[] {
          ADDITION, SUBTRACTION, MULIPLICATION, DIVISION
          });
      }
    }
```