1) What is String in Java?

Ans:
A string in Java is like a bunch of characters put together to represent some text.
It can be anything from a simple word, a sentence, or even a long paragraph.

Example,

String message = "Hello, World!";

The string variable `message` now holds the text "Hello, World!".

Strings in Java are special because they cannot be changed once they are created. This means that if we want to modify a string, we actually create a new string with the modified value. This property of strings is called immutability.

2) Types of String in Java are?

Ans:
Types of strings:

1. Immutable Strings:
   - Immutable strings are instances of the `String` class that cannot be changed once created.
   - String literals, created using double quotes, are immutable by default.
   - Immutable strings are thread-safe and can be safely shared among multiple threads.
   - Operations on immutable strings, such as concatenation or substring, create new string objects rather than modifying the existing ones.

2. Mutable Strings:
   - Mutable strings can be modified after creation.
   - Java provides a mutable string class called `StringBuilder` that allows efficient manipulation of strings.
   - Mutable strings are useful when there is a need for frequent modifications to the string content, such as concatenating multiple strings or building a string dynamically.
   - Unlike immutable strings, mutable strings are not thread-safe.

3) In how many ways can you create string object in java?

Ans:
In Java, there are three classes that can be used to create a `String` object:

1. String Class:
   - The `String` class itself can be used to create a `String` object.
   - String objects created using this class are immutable, meaning their values cannot be changed once created

Example:
```
  // Using String class
        String str1 = "Hello";
 // Using new keyword
        String str2 = new String("Java");
```

2. StringBuilder Class:
   - The `StringBuilder` class provides a mutable sequence of characters.
   - It can be used to efficiently construct and modify strings.
   - StringBuilder objects are mutable, allowing you to modify their content.

Example:
```
// Using StringBuilder class
StringBuilder sb = new StringBuilder(“Welcome”);
```

3. StringBuffer Class:
   - The `StringBuffer` class is similar to `StringBuilder` and provides a mutable sequence of characters.
   - StringBuffer objects are also used to construct and modify strings.

Example

```
// Using StringBuffer class
StringBuffer stringBuffer = new StringBuffer();
```

4) What is string constant pool?

Ans:

- The String Constant Pool is a special area in the Java heap memory.

- It is used to store string literals, which are strings created using double quotes.

- Strings in the pool are shared and reused, saving memory and improving performance.

- When a string literal is encountered, the JVM checks if an identical string exists in the pool.

- If a matching string is found, the new reference points to the existing string in the pool.

- If a matching string is not found, a new string is created in the pool and the reference points to it.

- Strings in the pool are immutable and cannot be changed. They an be used to explicitly add a string to the pool or obtain its reference from the pool.

- The String Constant Pool is specific to each JVM instance and not shared across different instances or applications.

5) What do you mean by mutable and immutable objects?

1. Mutable Objects:
   - Mutable objects are those whose state can be modified after creation.
   - Changes can be made to the object's fields or properties, altering its internal state.
   - Examples of mutable objects in Java include `StringBuilder`, `ArrayList`, and `HashMap`.
   - Mutable objects are generally not thread-safe, meaning they may encounter issues in concurrent or multi-threaded environments if not properly synchronized.

Example of a mutable object in Java:

```
public class MutablePerson {
    private String name;

    public MutablePerson(String name) {
        this.name = name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

In the `MutablePerson` class above, the `name` field can be modified by calling the `setName()` method. The object's state can change, allowing the name to be updated.

2. Immutable Objects:
   - Immutable objects are those whose state cannot be changed after creation.
   - Once created, an immutable object's fields or properties cannot be modified.
   - Examples of immutable objects in Java include `String`, `Integer`, and `LocalDate`.
   - Immutable objects are inherently thread-safe, as they cannot be modified and shared across threads.

Example of an immutable object in Java:

```
public class ImmutablePerson {
    private final String name;

    public ImmutablePerson(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}
```

In the `ImmutablePerson` class above, the `name` field is declared as `final`, indicating that its value cannot be changed after initialization. There are no setter methods, and the state of the object remains constant once created.

6) Where exactly is the string constant pool located in the memory?

Ans:
string constant pool is a special memory area in the heap where the values of all the strings which are defined in the program are stored. When we declare a String literal, the JVM creates the object in the pool and stores its reference on the stack. Before creating each String object in memory, the JVM performs some steps to decrease the memory overhead. The String constant pool uses a Hashmap in its implementation. Each bucket of the Hashmap contains a list of Strings with the same hash code.