Java Interview Question :→

(1) String, StringBuffer and StringBuilder :→

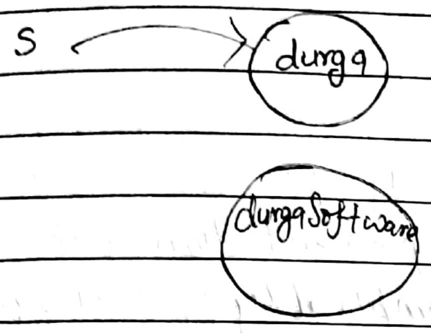
String

(1) String is a immutable Object

String s = new String("durga");

s. concat("software");

s.op(s) ⇒ durga ✓



→once we creates a String Object we can't perform any changes in the existing objects. If we are trying to perform any changes with those changes, a new String object will be created. This non changeble nature is nothing but immutability of the String object

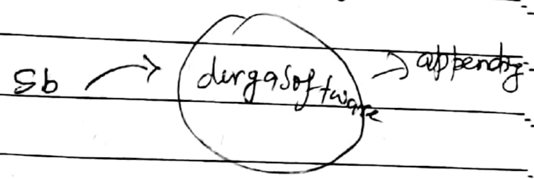
StringBuffer

(1) StringBuffer is mutable Object.

~~sb~~ String Buffer sb = new StringBuffer("durga");

sb.append("software");

s.o.p(sb); → durgasoftware



→once we creates a string Buffer objects we can perform any types of changes in the existing Object. This changeble is nothing but mutability of the StringBuffer Object.

Note :-

(1) If the content is fixed and won't change frequently then we should go for String.

(2) If the content is not fixed and keep on changing but Thread Safety is required. then we should go for StringBuffer.

(3) If the content is not fixed and keep on changing and thread safety is not required then we should go for StringBuffer.

→ StringBuilder is exactly same as StringBuffer (including Meth and constructors) except the following differences:

Page No.:

StringBuffer	StringBuilder
① Every method is Synchronized	① ~~Every~~ Non - Synchronized
② Thread Safe	② not Thread Safe.
③ ~~Introduced~~ Performance is Low	③ performance is fast
④ Introduced in 1.0 V.	④ Introduced in 1.5 V

Q② → Diff b/w Interface and Abstract class :-

Interface	Abstract class
① If we don't know anything about the implementation Just we have requirement specification then we Should gofor Interface.	② If we are talking about implementation but not compleatly (partialy) implementa then we Should go for Abstract class.
② Every method is always public and abstract whether we are declaring or not. Hence, Interface is also concidered as 100% pure Abstract class.	② Every method present in abstract class need not be public and Abstract. In addition to abstract methods, we can take concrete method also.
③ We can't declare interface method with the following modifiers :— ✓ public ~~method~~ → private, ~~protected~~ ; Abstract → final, Static, Synchronized, native, Strictfp.	③ There are no restriction on Abstract class method modifier.
④ Every variable present inside interface is always public, Static and final whether we are declaring on not.	④ The variable present inside Abstract class need not be public, Static and final.

Scanned by CamScanner