# Java APIs – Answers

Question

h None of the above

Strings are immutable. No methods of the String class will change the contents of a String instance. Some String methods such as concat, replace, substring and trim will return a new String with the desired modifications. The StringBuffer and StringBuilder class has methods to append, delete and insert – but these are NOT members of the java.lang.String class. A typical trick question will attempt to invoke StringBuffer or StringBuilder methods on a String instance.

Question

b Prints: A B C

The String instance referenced by s2 is passed to the m1 method by passing the value of the reference. The reference value used in method m1 is a local copy of the reference. If the local copy used in method m1 is changed, then the original reference variable in the main method remains unchanged.

Question

**d Prints: AZA**

Instances of type String are immutable. In method m1, the replace method returns a new instance of type String that contains the value Y, but the String instance referenced by s1 remains unchanged. The original value, A, is printed in method m1. In method m2, the replace method returns a new instance of type String that contains the value Z, and a reference to the new instance is assigned to reference variable s1. The new value, Z, is printed in method m2. In the main method, a copy of the reference value contained by the reference variable s1 is passed as an argument to methods m1 and m2. Since String instances are immutable, methods m1 and m2 can not change the original String instance that is declared in the main method. Since references are passed by value, methods m1 and m2 can not change the reference variable declared in the main method. Regardless of anything that happens in methods m1 and m2, the reference variable s1 that is declared in the main method will continue to reference the original String instance that contains the value A.

Question

d Prints: "ABCDEF"

The reference variable s1 is initialized with a reference to an instance of type String containing the value "ABCDEFG". The expression s2 = s1.substring(0,3) initializes the reference variable s2 with a reference to a unique instance of type String containing the value "ABC". The expression s3 = s1.substring(4,6) creates a unique instance of type String containing the value "EF". The expression c1 = s1.charAt(3) initializes the primitive variable c1 with the value 'D'. The expression String.valueOf(c1) invokes the static valueOf method with an argument of type primitive char and value 'D'. The valueOf method creates a new instance of type String. The value contained by the new instance is "D". The expression s2.concat(String.valueOf(c1)) invokes the concat method on the instance of type String referenced by the variable s2. The instance referenced by s2 contains the value "ABC". The value contained by the argument is "D". The result of the concatenation operation is a new instance of type String containing the value "ABCD". The expression s2.concat(String.valueOf(c1)).concat(s3) invokes the concat method on the previously created instance containing the value "ABCD". The instance referenced by the argument s3 contains the value "EF". The result of the concatenation operation is a new instance of type String containing the value "ABCDEF".

Question

**f Prints: ABABCABC**

Instances of type StringBuilder are not immutable. In method m1, the method invocation expression s1.append("B") appends the String literal "B" to the StringBuilder instance referenced by variable s1. The append method returns a reference to the same StringBuilder instance on which it is invoked; so the assignment expression s1 = s1.append("B") does not assign a different reference value to variable s1. The new value, AB, is printed in method m1. In method m2, the method invocation expression s1.append("C") appends the String literal "C" to the StringBuilder instance referenced by variable s1. The new value, ABC, is printed in method m2. In the main method, a copy of the reference value contained by the reference variable s1 is passed as an argument to methods m1 and m2. Since StringBuilder instances are not immutable, methods m1 and m2 are able to change the original StringBuilder instance that is declared in the main method. The new value, ABC, is printed in the main method.

Question 6

d

Compilation and output of the serialized content of the obs String array to a new file called mystrings.ob. Arrays are objects, and implement Serializable. Provided the array contents are Serializable, there is no problem. String is Serializable!