Q01:Which of the following statements are true ?

* The + operator is overloaded for concatenation for the String class
* Strings are a primitive type in Java and the StringBuffer is used as the matching wrapper type
* The size of a string can be retrieved using the length

property

Q02:Which two scenarios are NOT safe to replace a StringBuffer

object with a StringBuilder object? (Choose two.)

A. When using versions of Java technology earlier than 5.0.

B. When sharing a StringBuffer among multiple threads.

C. When using the java.io class StringBufferInputStream.

D. When you plan to reuse the StringBuffer to build more

than one string.

Q03:Given:

1. public class KungFu {

2. public static void main(String[] args) {

3. Integer x = 400;

4. Integer y = x;

5. x++;

6. StringBuilder sb1 = new StringBuilder("123");

7. StringBuilder sb2 = sb1;

8. sb1.append("5");

9. System.out.println((x==y) + " " + (sb1==sb2));

10. }

11. }

What is the result?

A. true true

B. false true

C. true false

D. false false

E. Compilation fails.

F. An exception is thrown at runtime.

:false true

Q04:Given:

1. public class TestString3 {

2. public static void main(String[] args) {

3. // insert code here

5. System.out.println(s);

6. }

7. }

Which two code fragments, inserted independently at line 3,

generate the output 4247?(Choose two.)

A. String s = "123456789";

s = (s-"123").replace(1,3,"24") - "89";

B. StringBuffer s = new StringBuffer("123456789");

s.delete(0,3).replace(1,3,"24").delete(4,6);

C. StringBuffer s = new StringBuffer("123456789");

s.substring(3,6).delete(1,3).insert(1, "24");

D. StringBuilder s = new StringBuilder("123456789");

s.substring(3,6).delete(1,2).insert(1, "24");

E. StringBuilder s = new StringBuilder("123456789");

s.delete(0,3).delete(1,3).delete(2,5).insert(1, "24");

Q05:Given:

22. StringBuilder sb1 = new StringBuilder("123");

23. String s1 = "123";

24. // insert code here

25. System.out.println(sb1 + " " + s1);

Which code fragment, inserted at line 24, outputs "123abc

123abc"?

A. sb1.append("abc"); s1.append("abc");

B. sb1.append("abc"); s1.concat("abc");

C. sb1.concat("abc"); s1.append("abc");

D. sb1.concat("abc"); s1.concat("abc");

E. sb1.append("abc"); s1 = s1.concat("abc");

F. sb1.concat("abc"); s1 = s1.concat("abc");

G. sb1.append("abc"); s1 = s1 + s1.concat("abc");

H. sb1.concat("abc"); s1 = s1 + s1.concat("abc");

Q06:Given:

1. public class TestString1 {

2. public static void main(String[] args) {

3. String str = "420";

4. str += 42;

5. System.out.print(str);

6. }

7. }

What is the output?

A. 42

B. 420

C. 462

D. 42042

E. Compilation fails.

F. An exception is thrown at runtime.

Q07:How many String objects are created?

public class MainClass {

public static void main(String[] argv) {

String A, B, C;

A = new String("1234");

B = A;

C = A + B;

}

}

A. One

B. Two

C. Three

D. Four

Q08:The StringBuffer class is used for strings that are not

allowed to change. The String class is used for strings that   
 are modified by the program: True or False. If false,explain

why.

A. False.

Q09:The use of the new operator is required for instantiation of

objects of type String: True or False? If false, explain your

answer.

ANS: False. A String object can be instantiated using either

of the following statements:

String str1 = new String("String named str2");

String str2 = "String named str1";

Q10:The use of the new operator is required for instantiation of

objects of type StringBuffer: True or False? If false,

explain your answer.

A - True.

Q11:Without specifying any explicit numeric values, provide a

code fragment that will instantiate an empty StringBuffer

object of the correct initial length to contain the string

"StringBuffer named str6" and then store that string in the

object.

A. See the following code fragment:

StringBuffer str6 = new StringBuffer("StringBuffer named

str6".length());

str6.append("StringBuffer named str6");

Q12:Provide a code fragment consisting of a single statement

showing how to use the Integer wrapper class to convert a   
 string containing digits to an integer and store it in a

variable of type int.

A. See code fragment below

int num = new Integer("3625").intValue();

Q13:Explain the difference between the capacity() method and the

length() methods of the StringBuffer class.

* The capacity() method returns the amount of space

currently allocated for the StringBuffer object. The

length() method returns the amount of space used.

Q14:The following is a valid code fragment: True or False? If

false, explain why.

StringBuffer str6 = new StringBuffer("StringBuffer named

str6".length());

A - True.

Q15:Which of the following code fragments is the most efficient,

first or second?

String str1 = "THIS STRING IS NAMED str1";

String str1 = new String("THIS STRING IS NAMED str1");

A. The first code fragment is the most efficient.

Q16:Given:

1. public class TestString3 {

2. public static void main(String[] args) {

3. // insert code here

5. System.out.println(s);

6. }

7. }

Which two code fragments, inserted independently at line 3,

generate the output 4247?(Choose two.)

A. String s = "123456789";

s = (s-"123").replace(1,3,"24") - "89";

B. StringBuffer s = new StringBuffer("123456789");

s.delete(0,3).replace(1,3,"24").delete(4,6);

C. StringBuffer s = new StringBuffer("123456789");

s.substring(3,6).delete(1,3).insert(1, "24");

D. StringBuilder s = new StringBuilder("123456789");

s.substring(3,6).delete(1,2).insert(1, "24");

E. StringBuilder s = new StringBuilder("123456789");

s.delete(0,3).delete(1,3).delete(2,5).insert(1, "24");

Q17:Given:

A. d is a valid, non-null Date object

B. df is a valid, non-null DateFormat object set to the

current locale What outputs the current locales country

name and the appropriate version of d’s date?

A. Locale loc = Locale.getLocale();

System.out.println(loc.getDisplayCountry()+ “ “+

df.format(d));

B. Locale loc = Locale.getDefault();

System.out.println(loc.getDisplayCountry()+ “ “ +

df.format(d));

C. Locale bc = Locale.getLocale();

System.out.println(loc.getDisplayCountry()+ “ “+

df.setDateFormat(d));

D. Locale loc = Locale.getDefault();

System.out.println(loc.getDispbayCountry()+ “ “+

df.setDateFormat(d));