1. Dado o programa abaixo, responda aos questionamentos a seguir, ao final da execução:

**import** java.lang.reflect.Field;

**import** sun.misc.Unsafe;

**public** **class** Exercicio01 {

**static** **final** Unsafe ***unsafe*** = *getUnsafe*();

**static** **final** **boolean** ***is64bit*** = **true**;

**public** **static** **void** printAddresses(String label, Object... objects) {

System.***out***.print(label + ": 0x");

**long** last = 0;

**long** offset = ***unsafe***.arrayBaseOffset(objects.getClass());

**int** scale = ***unsafe***.arrayIndexScale(objects.getClass());

**switch** (scale) {

**case** 4:

**long** factor = ***is64bit*** ? 8 : 1;

**final** **long** i1 = (***unsafe***.getInt(objects, offset) & 0xFFFFFFFFL) \* factor;

System.***out***.print(Long.*toHexString*(i1));

last = i1;

**for** (**int** i = 1; i < objects.length; i++) {

**final** **long** i2 = (***unsafe***.getInt(objects, offset + i \* 4) & 0xFFFFFFFFL) \* factor;

**if** (i2 > last)

System.***out***.print(", +" + Long.*toHexString*(i2 - last));

**else**

System.***out***.print(", -" + Long.*toHexString*( last - i2));

last = i2;

}

**break**;

**case** 8:

**throw** **new** AssertionError("Not supported");

}

System.***out***.println();

}

**private** **static** Unsafe getUnsafe() {

**try** {

Field theUnsafe = Unsafe.**class**.getDeclaredField("theUnsafe");

theUnsafe.setAccessible(**true**);

**return** (Unsafe) theUnsafe.get(**null**);

} **catch** (Exception e) {

**throw** **new** AssertionError(e);

}

}

**public** **static** **void** main(String[] args) {

String string = "";

**int** z = 1;

*printAddresses*(z + ": string", string);

**for** (**int** i = 0; i < 12; i++) {

string += "string";

z++;

*printAddresses*(z + ": string", string);

}

z = 1;

StringBuilder sb = **new** StringBuilder("");

*printAddresses*(z + ": sb", sb);

**for** (**int** i = 0; i < 12; i++) {

sb.append("teste");

z++;

*printAddresses*(z + ": sb", sb);

}

z = 1;

StringBuffer sbf = **new** StringBuffer("");

*printAddresses*(z + ": sbf", sbf);

**for** (**int** i = 0; i < 12; i++) {

sbf.append("teste");

z++;

*printAddresses*(z + ": sbf", sbf);

}

}

}

1. Foram criados 13 objetos
2. Strings são imutáveis e são armazenadas em um pool. Para cada concatenação, um novo objeto é criado, logo, se alterarmos para StringBuffer (controle de concorrência) ou StringBuilder (sem controle de concorrência), trabalharemos com uma instância desse objeto, já que ele foi criado fora do laço, e depois invocamos o append apenas para as novas partes da String, sem ter de copiar o que já foi previamente processado