100％合格率の1z1-809日本語版問題解説 &合格スムーズ1z1-809日本語的中対策 |有難い1z1-809受験資格Java SE 8 Programmer II

私たちの1z1-809試験参考書を利用し、1z1-809試験に合格できます。おそらくあなたは私たちの1z1-809試験参考書を信じられないでしょう。でも、あなたは1z1-809試験参考書を買ったお客様のコメントを見ると、すぐ信じるようになります。あなたは心配する必要がないです。早く1z1-809試験参考書を買いましょう!

1z1-809 Java SE 8 Programmer II試験の準備には、Javaプログラミング言語の概念とJava SE 8プラットフォームの包括的な理解が必要です。この認定試験は、効率的で最新のJava機能を取り入れたJavaコードを書き、デバッグ、テスト、トラブルシューティングできる候補者の能力を評価するように設計されています。さらに、この試験ではJava APIの知識とそのアプリケーション作成における使用方法もテストされます。Oracleは、試験のための候補者に、Java SE 8プログラミングの基本的なスキルの知識とJavaを使用してアプリケーションを開発する実践的な経験を持っていることを推奨しています。

Java SE 8のOracle Certified Professional（OCP）になるには、候補者がOCAとOCP試験の両方に合格する必要があります。 1Z1-809試験は、スキルを向上させ、Java SE 8テクノロジーの専門知識を実証したい経験豊富なJava開発者を対象としています。この試験は、候補者の知識と実践的なスキルをテストするために、複数選択の質問、ドラッグアンドドロップの質問、シナリオベースの質問で構成されています。

>> 1z1-809日本語版問題解説 <<

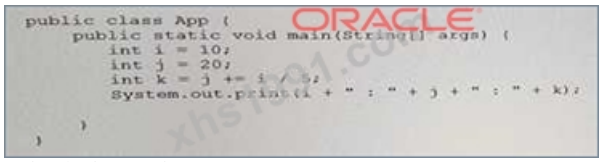
Oracle 1z1-809日本語的中対策、1z1-809受験資格

Xhs1991のサイトは長い歴史を持っていて、Oracleの1z1-809認定試験の学習教材を提供するサイトです。長年の努力を通じて、Xhs1991のOracleの1z1-809認定試験の合格率が１００パーセントになっていました。Oracleの1z1-809試験トレーニング資料の高い正確率を保証するために、うちはOracleの1z1-809問題集を絶えずに更新しています。それに、うちの学習教材を購入したら、私たちは一年間で無料更新サービスを提供することができます。

Java SE 8 Programmer II 認定試験は、Javaプログラミングに関する候補者の知識をテストする複数選択問題の試験であり、Java Class Design、Advanced Java Class Design、Generics and Collections、Lambda Built-in Functional Interfaces、Java Stream API、Exceptions and Assertions、Java I/O Fundamentals、Java Concurrencyを含みます。この試験は、150分以内に80個の複数選択問題を解答する必要があります。

Oracle Java SE 8 Programmer II 認定 1z1-809 試験問題 (Q20-Q25):

質問 # 20
Given:



What is the result?

- A. 10 : 22 : 6
- B. 10 : 22 : 20
- C. 10 : 22 : 22
- D. 10 : 30 : 6

正解：C

解説：
i = 10, so 10 / 5 = 2
j = 20 but 2 is added (from i) so j = 22
k = j = 22

質問 # 21
Given:

And given the commands:

What is the result?

- A. true true
- B. TRUE null
- C. true false
- D. false false
- E. A ClassCastException is thrown at runtime.

正解：C

質問 # 22
Given:
1 . abstract class Shape {
2 . Shape ( ) { System.out.println ("Shape"); }
3 . protected void area ( ) { System.out.println ("Shape"); }
4 . }
5 .
6 . class Square extends Shape {
7 . int side;
8 . Square int side {
9 . /* insert code here */
1 0. this.side = side;
1 1. }
1 2. public void area ( ) { System.out.println ("Square"); }
1 3. }
1 4. class Rectangle extends Square {
1 5. int len, br;
1 6. Rectangle (int x, int y) {
1 7. /* insert code here */
1 8. len = x, br = y;
1 9. }

2 0. void area ( ) { System.out.println ("Rectangle"); }
2 1. }
Which two modifications enable the code to compile?

- A. At line 1, remove abstract
- B. At line 12, remove public
- C. At line 17, insert super (); super.side = x;
- D. At line 17, insert super (x);
- E. At line 20, use public void area ( ) {
- F. At line 9, insert super ( );

正解：B、D

質問 # 23
Given:

```
class Equal {
    public static void main(String[] args){
    String str1 = "Java";
    String[] str2 = {"J", "a", "v", "a"};
    String str3 = "";
    for(String str : str2) {
        str3= str3+str;
    }
    boolean b1 = (str1 == str3);
    boolean b2 = (str1.equals(str3));
    System.out.print(b1+", "+b2);
  }
}
```

What is the result?

- A. true, true
- B. false, false
- C. false, true
- D. true, false

正解：C

質問 # 24
Given:

```
interface Calculator {
    public int increase(int x);
}

public class App {
    public static void main(String[] args) {
        Calculator c = new Calculator() {
                            public int increase(int x) {
                                return x + 100;
                            }
                        };
        int x = c.increase(1000);
        System.out.println(x);
    }
}
```

Which is refactored code with functional interfaces?

- A.

```
BiFunction<Integer, Integer> f = n -> n + 100;
int y = f.accept(1000);
System.out.println(x);
```

- B.
```
Function<Integer, Integer> f = n -> n + 100;
int y = f.apply(1000);
System.out.println(x);
```
- C.
```
Function<Integer> f = n -> n + 100;
int y = f.accept(1000);
System.out.println(x);
```
- D.
```
BiFunction<Integer, Integer> f = n -> n + 100;
int y = f.apply(1000);
System.out.println(x);
```

正解：B

質問 # 25

.....