

Java Package

For more details on SUN Certifications, visit [JavaScjpDumps](http://JavaScjpDumps.com)

Q:01 Given:

```
public class Person {  
    private String name, comment;  
    private int age;  
    public Person(String n, int a, String c) {  
        name = n; age = a; comment = c;  
    }  
    public boolean equals(Object o) {  
        if (! (o instanceof Person)) return false;  
        Person p = (Person)o;  
        return age == p.age && name.equals(p.name);  
    }  
}
```

What is the appropriate definition of the hashCode method in class Person?

- A. return super.hashCode();
- B. return name.hashCode() + age * 7;
- C. return name.hashCode() + comment.hashCode() / 2;
- D. return name.hashCode() + comment.hashCode() / 2 - age * 3;

Answer: B

Q: 02 Given this method in a class:

```
21. public String toString() {  
22.     StringBuffer buffer = new StringBuffer();  
23.     buffer.append('<');  
24.     buffer.append(this.name);  
25.     buffer.append('>');  
26.     return buffer.toString();  
27. }
```

Which statement is true?

- A. This code is NOT thread-safe.
- B. The programmer can replace StringBuffer with StringBuilder with no other changes.
- C. This code will perform poorly. For better performance, the code should be rewritten:
return "<" + this.name + ">"
- D. This code will perform well and converting the code to use StringBuilder will not enhance the performance.

Answer: B

Q: 03 Given:

```
11. public void testIfA() {  
12. if (testIfB("True")) {  
13. System.out.println("True");  
14. } else {  
15. System.out.println("Not true");  
16. }  
17. }  
18. public Boolean testIfB(String str) {  
19. return Boolean.valueOf(str);  
20. }
```

What is the result when method testIfA is invoked?

- A. True
- B. Not true
- C. An exception is thrown at runtime.
- D. Compilation fails because of an error at line 12.
- E. Compilation fails because of an error at line 19.

Answer: A

Q: 04 Given:

```
1. public class Boxer1{  
2. Integer i;  
3. int x;  
4. public Boxer1(int y) {  
5. x = i+y;  
6. System.out.println(x);  
7. }  
8. public static void main(String[] args) {  
9. new Boxer1(new Integer(4));  
10. }  
11. }
```

What is the result?

- A. The value "4" is printed at the command line.
- B. Compilation fails because of an error in line 5.
- C. Compilation fails because of an error in line 9.
- D. A NullPointerException occurs at runtime.
- E. A NumberFormatException occurs at runtime.
- F. An IllegalStateException occurs at runtime.

Answer: D

Q: 05 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");

Answer: B, E

Q: 06 Given:

11. public static void test(String str) {

12. int check = 4;

13. if (check == str.length()) {

14. System.out.print(str.charAt(check - 1) + ", ");

15. } else {

16. System.out.print(str.charAt(0) + ", ");

17. }

18. }

and the invocation:

21. test("four");

22. test("tee");

23. test("to");

What is the result?

A. r, t, t,

B. r, e, o,

C. Compilation fails.

D. An exception is thrown at runtime.

Answer: C

Q: 07 Given:

11. public class Person {

12. private String name;

13. public Person(String name) {

14. this.name = name;

```

15. }
16. public boolean equals(Object o) {
17. if ( ! o instanceof Person ) return false;
18. Person p = (Person) o;
19. return p.name.equals(this.name);
20. }
21. }

```

Which statement is true?

- A. Compilation fails because the hashCode method is not overridden.
- B. A HashSet could contain multiple Person objects with the same name.
- C. All Person objects will have the same hash code because the hashCode method is not overridden.
- D. If a HashSet contains more than one Person object with name="Fred", then removing another Person, also with name="Fred", will remove them all.

Answer: B

Q: 08 Which two statements are true about the hashCode method? (Choose two.)

- A. The hashCode method for a given class can be used to test for object equality and object inequality for that class.
- B. The hashCode method is used by the java.util.SortedSet collection class to order the elements within that set.
- C. The hashCode method for a given class can be used to test for object inequality, but NOT object equality, for that class.
- D. The only important characteristic of the values returned by a hashCode method is that the distribution of values must follow a Gaussian distribution.
- E. The hashCode method is used by the java.util.HashSet collection class to group the elements within that set into hash buckets for swift retrieval.

Answer: C, E

Q: 09 Click the Task button.

Place the code into the GenericB class definition to make the class compile successfully.

```
import java.util.*;
```

```
public class GenericB<Place> {  
    public Place foo;  
    public void setFoo(Place foo) {  
        this.foo = foo;  
    }  
    public Place getFoo() {  
        return foo;  
    }  
    public static void main (String[] args) {  
        GenericB<Cat> bar = new GenericB<Cat>();  
        bar.setFoo(new Cat());  
        Cat c = bar.getFoo();  
    }  
}
```

```
interface Pet { }  
class Cat implements Pet{ }
```

Code

? extends Pet

T extends Pet

? implements Pet

T implements Pet

Pet extends T

?

T

<?>

Pet

Done

1.<T extends Pet>

2. T

3.T

4.T

Q: 10 Given:

10. public class MyClass {

11.

12. public Integer startingI;

13. public void methodA() {

14. Integer i = new Integer(25);

15. startingI = i;

16. methodB(i);

17. }

18. private void methodB(Integer i2) {

19. `i2 = i2.intValue();`

20.

21. `}`

22. `}`

If methodA is invoked, which two are true at line 20? (Choose two.)

A. `i2 == startingI` returns true.

B. `i2 == startingI` returns false.

C. `i2.equals(startingI)` returns true.

D. `i2.equals(startingI)` returns false.

Answer: B, C

Question: 11

Given:

11. `public String makinStrings() {`

12. `String s = "Fred";`

13. `s = s + "47";`

14. `s = s.substring(2, 5);`

15. `s = s.toUpperCase();`

16. `return s.toString();`

17. `}`

How many String objects will be created when this method is invoked?

A. 1

B. 2

C. 3

D. 4

E. 5

F. 6

Answer: E

12. Which statements are true about comparing two instances of the same class, given that the `equals()` and `hashCode()` methods have been properly overridden? (Choose all that apply.)

A. If the `equals()` method returns true, the `hashCode()` comparison `==` might return false.

B. If the `equals()` method returns false, the `hashCode()` comparison `==` might return true.

C. If the `hashCode()` comparison `==` returns true, the `equals()` method must return true.

D. If the `hashCode()` comparison `==` returns true, the `equals()` method might return true.

E. If the `hashCode()` comparison `!=` returns true, the `equals()` method might return true.

Answer:

- > **B** and **D**. **B** is true because often two dissimilar objects can return the same hashcode value.

D is true because if the `hashCode()` comparison returns `==`, the two objects might or might not be equal.

-> **A**, **C**, and **E** are incorrect. **C** is incorrect because the `hashCode()` method is very flexible in its return values, and often two dissimilar objects can return the same hash code value. **A** and **E** are a negation of the `hashCode()` and `equals()` contract.

13. Given:

```

1. class Convert {
2. public static void main(String[] args) {
3. Long xL = new Long(456L);
4. long x1 = Long.valueOf("123");
5. Long x2 = Long.valueOf("123");
6. long x3 = xL.longValue();
7. Long x4 = xL.longValue();
8. Long x5 = Long.parseLong("456");
9. long x6 = Long.parseLong("123");
10. }
11. }

```

Which will compile using Java 5, but will NOT compile using Java 1.4? (Choose all that apply.)

- A. Line 4.
- B. Line 5.
- C. Line 6.
- D. Line 7.
- E. Line 8.
- F. Line 9.

Answer:

-> **A, D,** and **E** are correct. Because of the methods' return types, these method calls required autoboxing to compile.

-> **B, C,** and **F** are incorrect based on the above.

14. Given:

```

class TKO {
public static void main(String[] args) {
String s = "-";
Integer x = 343;
long L343 = 343L;
if(x.equals(L343)) s += ".e1 ";
if(x.equals(343)) s += ".e2 ";
Short s1 = (short)((new Short((short)343)) / (new Short((short)49)));
if(s1 == 7) s += "=s ";
if(s1 < new Integer(7+1)) s += "fly ";
System.out.println(s);
}}

```

Which of the following will be included in the output String s? (Choose all that apply.)

- A. .e1
- B. .e2
- C. =s
- D. fly
- E. None of the above.
- F. Compilation fails.
- G. An exception is thrown at runtime.

Answer:

-> **B**, **C**, and **D** are correct. Remember, that the equals() method for the integer wrappers will only return true if the two primitive types and the two values are equal. With **C**, it's okay to unbox and use ==. For **D**, it's okay to create a wrapper object with an expression, and unbox it for comparison with a primitive.

-> **A**, **E**, **F**, and **G** are incorrect based on the above. (Remember that **A** is using the equals() method to try to compare two different types.)

15. Which about the three java.lang classes String, StringBuilder, and StringBuffer are true? (Choose all that apply.)

- A. All three classes have a length() method.
- B. Objects of type StringBuffer are thread-safe.
- C. All three classes have overloaded append() methods.
- D. The "+" is an overloaded operator for all three classes.
- E. According to the API, StringBuffer will be faster than StringBuilder under most implementations.
- F. The value of an instance of any of these three types can be modified through various methods in the API.

Answer:

-> **A** and **B** are correct.

-> **C** is incorrect because String does not have an "append" method. **D** is incorrect because only String objects can be operated on using the overloaded "+" operator. **E** is backwards, StringBuilder is typically faster because it's not thread-safe. **F** is incorrect because String objects are immutable. A String reference can be altered to refer to a different String object, but the objects themselves are immutable.

16. Given:

```
class Polish {  
    public static void main(String[] args) {  
        int x = 4;  
        StringBuffer sb = new StringBuffer("..fedcba");  
        sb.delete(3,6);  
        sb.insert(3, "az");  
        if(sb.length() > 6) x = sb.indexOf("b");  
        sb.delete((x-3), (x-2));  
        System.out.println(sb);  
    }  
}
```

What is the result?

- A. .faza
- B. .fzba
- C. ..azba
- D. .fazba

E. ...fezba

F. Compilation fails.

G. An exception is thrown at runtime.

Answer:

-> **C** is correct. Remember that StringBuffer methods use zero-based indexes, and that ending indexes are typically exclusive.

-> **A, B, D, E, F,** and **G** are incorrect based on the above. (Objective 3.1)