# Test results by question

**Test:** JAVA TEST BANK 1 (ID# 6d511a7daa7c6455)

**Candidate:** AZEEM MAREDIYA

**Score** 0%

**Question 1 (ID: # 254770)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 02 |

An Instance of a Class, _____.

Which of these is false?

| | |
|---|---|
| A. | is an object |
| B. | Objects can access both static and instance data |
| C. | Object is the super class of all other classes |
| D. | Objects do not permit encapsulation |

Correct Answer D

User Answer

Elapsed Time (seconds) 10

Explanation Correct Answer d
Explanation
No Explanation

**Question 2 (ID: # 254771)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Language Fundamentals 2 |

Which will legally declare, construct, and initialize an array?

| | |
|---|---|
| A. | int [] myList = {"1", "2", "3"}; |
| B. | int [] myList = (5, 8, 2); |
| C. | int myList [] [] = {4,9,7,0}; |
| D. | int myList [] = {4, 3, 7}; |

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

The only legal array declaration and assignment statement is Option D

Option A is wrong because it initializes an int array with String literals.

Option B is wrong because it use something other than curly braces for the initialization.

Option C is wrong because it provides initial values for only one dimension, although the declared array is a two-dimensional array.

**Question 3 (ID: # 254772)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Language Fundamentals 3 |

Which is a valid keyword in java?

| | |
|---|---|
| A. | interface |
| B. | string |
| C. | Float |
| D. | unsigned |

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **A**

**Explanation:**

interface is a valid keyword.

Option B is wrong because although "String" is a class type in Java, "string" is not a keyword.

Option C is wrong because "Float" is a class type. The keyword for the Java primitive isfloat.

Option D is wrong because "unsigned" is a keyword in C/C++ but not in Java.

| **Question 4 (ID: # 254773)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 07 |

Which statement is true?

```
public class While
{
    public void loop()
    {
        int x= 0;
        while ( 1 ) /* Line 6 */
        {
            System.out.print("x plus one is " + (x + 1)); /* Line 8 */
        }
    }
}
```

A. There is a syntax error on line 1.
B. There are syntax errors on lines 1 and 6.
C. There are syntax errors on lines 1, 6, and 8.
D. There is a syntax error on line 6.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

Using the integer 1 in the while statement, or any other looping or conditional construct for that matter, will result in a compiler error. This is old C Program syntax, not valid Java.

A, B and C are incorrect because line 1 is valid (Java is case sensitive so While is a valid class name). Line 8 is also valid because an equation may be placed in a String operation as shown.

| **Question 5 (ID: # 254774)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 01 |

Which is the valid declarations within an interface definition?

| | |
|---|---|
| A. | public double methoda(); |
| B. | public final double methoda(); |
| C. | static void methoda(double d1); |
| D. | protected void methoda(double d1); |

| | |
|---|---|
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **A**

**Explanation:**

Option A is correct. A public access modifier is acceptable. The method prototypes in an interface are all abstract by virtue of their declaration, and should not be declaredabstract.

Option B is wrong. The final modifier means that this method cannot be constructed in a subclass. A final method cannot be abstract.

Option C is wrong. static is concerned with the class and not an instance.

Option D is wrong. protected is not permitted when declaring a method of an interface. See information below.

Member declarations in an interface disallow the use of some declaration modifiers; you cannot use transient, volatile, or synchronized in a member declaration in an interface. Also, you may not use the private and protected specifiers when declaring members of an interface.

**Question 6 (ID: # 254775)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 03 |

Which one is a valid declaration of a boolean?

A. boolean b1 = 0;
B. boolean b2 = 'false';
C. boolean b3 = false;
D. boolean b4 = Boolean.false();
E. boolean b5 = no;

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

A boolean can only be assigned the literal true or false.

**Question 7 (ID: # 254776)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 04 |

Which is a valid declarations of a String?

A. String s1 = null;
B. String s2 = 'null';
C. String s3 = (String) 'abc';
D. String s4 = (String) '\ufeed';

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **A**

**Explanation:**

Option A sets the String reference to null.

Option B is wrong because null cannot be in single quotes.

Option C is wrong because there are multiple characters between the single quotes ('abc').

Option D is wrong because you can't cast a char (primitive) to a String (object).

View Answer Workspace Report Discuss in Forum
14.

What is the numerical range of a char?

A.-128 to 127B.-(2$^{15}$) to (2$^{15}$) - 1C.0 to 32767D.0 to 65535

| **Question 8 (ID: # 254777)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 05 |

See Below.

```
public void foo( boolean a, boolean b)
{
    if( a )
    {
        System.out.println("A"); /* Line 5 */
    }
    else if(a && b) /* Line 7 */
    {
        System.out.println( "A && B");
    }
    else /* Line 11 */
    {
        if ( !b )
        {
            System.out.println( "notB") ;
        }
        else
        {
            System.out.println( "ELSE" ) ;
        }
    }
}
```

A. If a is true and b is true then the output is "A && B"

B. If a is true and b is false then the output is "notB"

C. If a is false and b is true then the output is "ELSE"

D. If a is false and b is false then the output is "ELSE"

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

Option C is correct. The output is "ELSE". Only when a is false do the output lines after 11 get some chance of executing.

Option A is wrong. The output is "A". When a is true, irrespective of the value of b, only the line 5 output will be executed. The condition at line 7 will never be evaluated (when a is true it will always be trapped by the line 12 condition) therefore the output will never be "A && B".

Option B is wrong. The output is "A". When a is true, irrespective of the value of b, only the line 5 output will be executed.

Option D is wrong. The output is "notB"

**Question 9 (ID: # 254778)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 06 |

Which statement is true?

```
public void test(int x)
{
    int odd = 1;
    if(odd) /* Line 4 */
    {
        System.out.println("odd");
    }
    else
    {
        System.out.println("even");
    }
}
```

A. Compilation fails.

B. "odd" will always be output.

C. "even" will always be output.

D. "odd" will be output for odd values of x, and "even" for even values.

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **A**

**Explanation:**

The compiler will complain because of incompatible types (line 4), the if expects a boolean but it gets an integer.

**Question 10 (ID: # 254779)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 08 |

Which three form part of correct array declarations?

1. public int a [ ]
2. static int [ ] a
3. public [ ] int a
4. private int a [3]
5. private int [3] a [ ]
6. public final int [ ] a

| | |
|---|---|
| A. | 1, 3, 4 |
| B. | 2, 4, 5 |
| C. | 1, 2, 6 |
| D. | 2, 5, 6 |
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **C**

**Explanation:**

(1), (2) and (6) are valid array declarations.

Option (3) is not a correct array declaration. The compiler complains with: illegal start of type. The brackets are in the wrong place. The following would work: public int[ ] a

Option (4) is not a correct array declaration. The compiler complains with: ']' expected. A closing bracket is expected in place of the 3. The following works: private int a []

Option (5) is not a correct array declaration. The compiler complains with 2 errors:

']' expected. A closing bracket is expected in place of the 3 and

expected A variable name is expected after a[ ] .

| **Question 11 (ID: # 254782)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |

| | |
|---|---|
| **Description** | Core Java 11 |

You want a class to have access to members of another class in the same package. Which is the most restrictive access that accomplishes this objective?

| | |
|---|---|
| A. | public |
| B. | private |
| C. | protected |
| D. | default access |
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **D**

**Explanation:**

The only two real contenders are C and D. Protected access Option C makes a member accessible only to classes in the same package or subclass of the class. While default access Option D makes a member accessible only to classes in the same package

| | |
|---|---|
| **Question 12 (ID: # 254783)** | |
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 12 |

Which is valid in a class that extends class A?

```
class A
{
    protected int method1(int a, int b)
    {
        return 0;
    }
}
```

| | |
|---|---|
| A. | public int method1(int a, int b) {return 0; } |
| B. | private int method1(int a, int b) { return 0; } |
| C. | public short method1(int a, int b) { return 0; } |
| D. | static protected int method1(int a, int b) { return 0; } |
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **A**

**Explanation:**

Option A is correct - because the class that extends A is just simply overriding method1.

Option B is wrong - because it can't override as there are less access privileges in the subclass method1.

Option C is wrong - because to override it, the return type needs to be an integer. The different return type means that the method is not overriding but the same argument list means that the method is not overloading. Conflict - compile time error.

Option D is wrong - because you can't override a method and make it a class method i.e. using static.

**Question 13 (ID: # 254784)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 09 |

What is the prototype of the default constructor?

```
public class Test { }
```

A. Test( )
B. Test(void)
C. public Test( )
D. public Test(void)

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

Option A and B are wrong because they use the default access modifier and the access modifier for the class is public (remember, the default constructor has the same access modifier as the class).

Option D is wrong. The void makes the compiler think that this is a method specification - in fact if it were a method specification the compiler would spit it out.

**Question 14 (ID: # 254785)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 10 |

Which three are valid method signatures in an interface?

1. private int getArea();
2. public float getVol(float x);
3. public void main(String [] args);
4. public static void main(String [] args);
5. boolean setFlag(Boolean [] test);

A. 1 and 2

B. 2, 3 and 5

C. 3, 4, and 5

D. 2 and 4

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **B**

**Explanation:**

(2), (3), and (5). These are all valid interface method signatures.

(1), is incorrect because an interface method must be public; if it is not explicitly declared public it will be made public implicitly. (4) is incorrect because interface methods cannot be static.

**Question 15 (ID: # 254786)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Language Fundamentals 1 |

Which four options describe the correct default values for array elements of the types indicated?

1. int -> 0
2. String -> "null"
3. Dog -> null

4. char -> '\u0000'
5. float -> 0.0f
6. boolean -> true

| | |
|---|---|
| A. | 1, 2, 3, 4 |
| B. | 1, 3, 4, 5 |
| C. | 2, 4, 5, 6 |
| D. | 3, 4, 5, 6 |

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **B**

**Explanation:**

(1), (3), (4), (5) are the correct statements.

(2) is wrong because the default value for a String (and any other object reference) isnull, with no quotes.

(6) is wrong because the default value for boolean elements is false.

**Question 16 (ID: # 254787)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 13 |

Which one creates an instance of an array?

| | |
|---|---|
| A. | int[ ] ia = new int[15]; |
| B. | float fa = new float[20]; |
| C. | char[ ] ca = "Some String"; |
| D. | int ia[ ] [ ] = { 4, 5, 6 }, { 1,2,3 }; |

| | |
|---|---|
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **A**

**Explanation:**

Option A is correct. It uses correct array declaration and correct array construction.

Option B is incorrect. It generates a compiler error: incompatible types because the array variable declaration is not correct. The array construction expects a reference type, but it is supplied with a primitive type in the declaration.

Option C is incorrect. It generates a compiler error: incompatible types because a string literal is not assignable to a character type variable.

Option D is wrong, it generates a compiler error expected. The compiler thinks that you are trying to create two arrays because there are two array initialisers to the right of the equals, whereas your intention was to create a 3 x 3 two-dimensional array.

| **Question 17 (ID: # 254788)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 14 |

What will be the output of the program?

```
class Test
{
    public static void main(String [] args)
    {
        Test p = new Test();
        p.start();
    }

    void start()
    {
        boolean b1 = false;
        boolean b2 = fix(b1);
        System.out.println(b1 + " " + b2);
    }

    boolean fix(boolean b1)
    {
        b1 = true;
        return b1;
    }
}
```

A. true true

B. false true

C. true false

D. false false

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **B**

**Explanation:**

The boolean b1 in the fix() method is a different boolean than the b1 in the start()method. The b1 in the start() method is not updated by the fix() method.

**Question 18 (ID: # 254789)**

**Subject** Core Java Level 1

**Subtopic** Java Test 1

**Description** Core Java 15

What will be the output of the program?

```
class PassS
{
    public static void main(String [] args)
    {
        PassS p = new PassS();
        p.start();
    }

    void start()
    {
        String s1 = "slip";
        String s2 = fix(s1);
        System.out.println(s1 + " " + s2);
    }

    String fix(String s1)
    {
        s1 = s1 + "stream";
        System.out.print(s1 + " ");
        return "stream";
    }
}
```

A. slip stream

B. slipstream stream

C. stream slip stream

D. slipstream slip stream

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **D**

**Explanation:**

When the fix() method is first entered, start()'s s1 and fix()'s s1 reference variables both refer to the same String object (with a value of "slip"). Fix()'s s1 is reassigned to a new object that is created when the concatenation occurs (this second String object has a value of "slipstream"). When the program returns to start(), another String object is created, referred to by s2 and with a value of "stream".

**Question 19 (ID: # 254790)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 16 |

What will be the output of the program?

```
class Equals
{
    public static void main(String [] args)
    {
        int x = 100;
        double y = 100.1;
        boolean b = (x = y); /* Line 7 */
        System.out.println(b);
    }
}
```

A. true
B. false
C. Compilation fails
D. An exception is thrown at runtime

| | |
|---|---|
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **C**

**Explanation:**

The code will not compile because in line 7, the line will work only if we use (x==y) in the line. The == operator compares values to produce a boolean, whereas the = operator assigns a value to variables.

Option A, B, and D are incorrect because the code does not get as far as compiling. If we corrected this code, the output would be false.

| **Question 20 (ID: # 254791)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 17 |

What will be the output of the program?

```
class Test
{
    public static void main(String [] args)
    {
        int x=20;
        String sup = (x < 15) ? "small" : (x < 22)? "tiny" : "huge";
        System.out.println(sup);
    }
}
```

A. small
B. tiny
C. huge
D. Compilation fails

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **B**

**Explanation:**

This is an example of a nested ternary operator. The second evaluation (x < 22) istrue, so the "tiny" value is assigned to sup.

**Question 21 (ID: # 254792)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 18 |

What will be the output of the program?

```
class Test
{
    public static void main(String [] args)
    {
        int x= 0;
        int y= 0;
        for (int z = 0; z < 5; z++)
        {
            if (( ++x > 2 ) && (++y > 2))
            {
                x++;
            }
        }
        System.out.println(x + " " + y);
    }
}
```

A. 5 2
B. 5 3
C. 6 3
D. 6 4

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

In the first two iterations x is incremented once and y is not because of the short circuit&& operator. In the third and forth iterations x and y are each incremented, and in the fifth iteration x is doubly incremented and y is incremented.

**Question 22 (ID: # 254793)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 19 |

What will be the output of the program?

```
class SC2
{
    public static void main(String [] args)
    {
        SC2 s = new SC2();
        s.start();
    }

    void start()
    {
        int a = 3;
        int b = 4;
        System.out.print(" " + 7 + 2 + " ");
        System.out.print(a + b);
        System.out.print(" " + a + b + " ");
        System.out.print(foo() + a + b + " ");
        System.out.println(a + b + foo());
    }

    String foo()
    {
        return "foo";
    }
}
```

A. 9 7 7 foo 7 7foo

B. 72 34 34 foo34 34foo

C. 9 7 7 foo34 34foo

D. 72 7 34 foo34 7foo

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

Because all of these expressions use the + operator, there is no precedence to worry about and all of the expressions will be evaluated from left to right. If either operand being evaluated is a String, the + operator will concatenate the two operands; if both operands are numeric, the + operator will add the two operands.

| **Question 23 (ID: # 254794)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 20 |

What will be the output of the program?

```
int i = 1, j = -1;
switch (i)
{
    case 0, 1: j = 1; /* Line 4 */
    case 2: j = 2;
    default: j = 0;
}
System.out.println("j = " + j);
```

A. j = -1
B. j = 0
C. j = 1
D. Compilation fails.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

The case statement takes only a single argument. The case statement on line 4 is given two arguments so the compiler complains.

**Question 24 (ID: # 254795)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 21 |

What will be the output of the program?

```
int i = 1, j = 10;
do
{
    if(i > j)
    {
        break;
    }
    j--;
} while (++i < 5);
System.out.println("i = " + i + " and j = " + j);
```

A. i = 6 and j = 5
B. i = 5 and j = 5

| | |
|---|---|
| C. | i = 6 and j = 4 |
| D. | i = 5 and j = 6 |
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **D**

**Explanation:**

This loop is a do-while loop, which always executes the code block within the block at least once, due to the testing condition being at the end of the loop, rather than at the beginning. This particular loop is exited prematurely if i becomes greater than j.

The order is, test i against j, if bigger, it breaks from the loop, decrements j by one, and then tests the loop condition, where a pre-incremented by one i is tested for being lower than 5. The test is at the end of the loop, so i can reach the value of 5 before it fails. So it goes, start:

1, 10

2, 9

3, 8

4, 7

5, 6 loop condition fails.

| **Question 25 (ID: # 254796)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 22 |

What will be the output of the program?

```
public class Switch2
{
    final static short x = 2;
    public static int y = 0;
    public static void main(String [] args)
    {
        for (int z=0; z < 3; z++)
        {
            switch (z)
            {
                case x: System.out.print("0 ");
                case x-1: System.out.print("1 ");
                case x-2: System.out.print("2 ");
            }
        }
    }
}
```

A. 0 1 2
B. 0 1 2 1 2 2
C. 2 1 0 1 0 0
D. 2 1 2 0 1 2

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

The case expressions are all legal because x is marked final, which means the expressions can be evaluated at compile time. In the first iteration of the for loop casex-2 matches, so 2 is printed. In the second iteration, x-1 is matched so 1 and 2 are printed (remember, once a match is found all remaining statements are executed until a break statement is encountered). In the third iteration, x is matched. So 0 1 and 2 are printed.

**Question 26 (ID: # 254797)**

**Subject** Core Java Level 1
**Subtopic** Java Test 1
**Description** Core Java 25

What will be the output of the program?

```
for(int i = 0; i < 3; i++)
{
    switch(i)
    {
        case 0: break;
        case 1: System.out.print("one ");
        case 2: System.out.print("two ");
        case 3: System.out.print("three ");
    }
}
System.out.println("done");
```

A. done
B. one two done
C. one two three done
D. one two three two three done

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

The variable i will have the values 0, 1 and 2.

When i is 0, nothing will be printed because of the break in case 0.

When i is 1, "one two three" will be output because case 1, case 2 and case 3 will be executed (they don't have break statements).

When i is 2, "two three" will be output because case 2 and case 3 will be executed (again no break statements).

Finally, when the for loop finishes "done" will be output.

**Question 27 (ID: # 254798)**

**Subject** Core Java Level 1
**Subtopic** Java Test 1
**Description** Core Java 23

What will be the output of the program?

```
for (int i = 0; i < 4; i += 2)
{
    System.out.print(i + " ");
}
System.out.println(i); /* Line 5 */
```

A. 0 2 4
B. 0 2 4 5
C. 0 1 2 3 4
D. Compilation fails.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

Compilation fails on the line 5 - System.out.println(i); as the variable i has only been declared within the for loop. It is not a recognised variable outside the code block of loop.

| **Question 28 (ID: # 254799)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 1 |
| **Description** | Core Java 24 |

What will be the output of the program?

```
int x = 3;
int y = 1;
if (x = y) /* Line 3 */
{
    System.out.println("x =" + x);
}
```

A. x = 1
B. x = 3
C. Compilation fails.
D. The code runs with no output.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

Line 3 uses an assignment as opposed to comparison. Because of this, the if statement receives an integer value instead of a boolean. And so the compilation fails.

**Question 29 (ID: # 254807)**

**Subject** Core Java Level 1
**Subtopic** Java Test 2
**Description** L2 Core Java 1

Which collection class allows you to grow or shrink its size and provides indexed access to its elements, but whose methods are not synchronized?

A. java.util.HashSet
B. java.util.LinkedHashSet
C. java.util.List
D. java.util.ArrayList

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **D**

**Explanation:**

All of the collection classes allow you to grow or shrink the size of your collection.ArrayList provides an index to its elements. The newer collection classes tend not to have synchronized methods. Vector is an older implementation of ArrayListfunctionality and has synchronized methods; it is slower than ArrayList.

**Question 30 (ID: # 254808)**

**Subject** Core Java Level 1
**Subtopic** Java Test 2
**Description** L2 Core Java 2

You need to store elements in a collection that guarantees that no duplicates are stored and all elements can be accessed in natural order . Which interface provides that capability?

| | |
|---|---|
| A. | java.util.Map |
| B. | java.util.Set |
| C. | java.util.List |
| D. | java.util.Collection |
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **B**

**Explanation:**

Option B is correct. A set is a collection that contains no duplicate elements. The iterator returns the elements in no particular order (unless this set is an instance of some class that provides a guarantee). A map cannot contain duplicate keys but it may contain duplicate values. List and Collection allow duplicate elements.

Option A is wrong. A map is an object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value. The Map interface provides three collection views, which allow a map's contents to be viewed as a set of keys, collection of values, or set of key-value mappings. The order of a map is defined as the order in which the iterators on the map's collection views return their elements. Some map implementations, like the TreeMap class, make specific guarantees as to their order (ascending key order); others, like the HashMap class, do not (does not guarantee that the order will remain constant over time).

Option C is wrong. A list is an ordered collection (also known as a sequence). The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list. Unlike sets, lists typically allow duplicate elements.

Option D is wrong. A collection is also known as a sequence. The user of this interface has precise control over where in the list each element is inserted. The user can access elements by their integer index (position in the list), and search for elements in the list. Unlike sets, lists typically allow duplicate elements.

**Question 31 (ID: # 254809)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 3 |

Which interface does java.util.Hashtable implement?

| | |
|---|---|
| A. | Java.util.Map |
| B. | Java.util.List |

| | |
|---|---|
| C. | Java.util.HashTable |
| D. | Java.util.Collection |
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **A**

**Explanation:**

Hash table based implementation of the Map interface.

| **Question 32 (ID: # 254810)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 4 |

Which interface provides the capability to store objects using a key-value pair?

| | |
|---|---|
| A. | Java.util.Map |
| B. | Java.util.Set |
| C. | Java.util.List |
| D. | Java.util.Collection |
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **A**

**Explanation:**

An object that maps keys to values. A map cannot contain duplicate keys; each key can map to at most one value.

| **Question 33 (ID: # 254811)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |

| | |
|---|---|
| **Description** | L2 Core Java 5 |

Which collection class allows you to associate its elements with key values, and allows you to retrieve objects in FIFO (first-in, first-out) sequence?

| | |
|---|---|
| A. | java.util.ArrayList |
| B. | java.util.LinkedHashMap |
| C. | java.util.HashMap |
| D. | java.util.TreeMap |

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **B**

**Explanation:**

LinkedHashMap is the collection class used for caching purposes. FIFO is another way to indicate caching behavior. To retrieve LinkedHashMap elements in cached order, use the values() method and iterate over the resultant collection.

| | |
|---|---|
| **Question 34 (ID: # 254812)** | |
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 6 |

Which collection class allows you to access its elements by associating a key with an element's value, and provides synchronization?

| | |
|---|---|
| A. | java.util.SortedMap |
| B. | java.util.TreeMap |
| C. | java.util.TreeSet |
| D. | java.util.Hashtable |

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **D**

**Explanation:**

Hashtable is the only class listed that provides synchronized methods. If you need synchronization great; otherwise, use HashMap, it's faster.

**Question 35 (ID: # 254813)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 7 |

Which is valid declaration of a float?

A. float f = 1F;
B. float f = 1.0;
C. float f = "1";
D. float f = 1.0d;

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **A**

**Explanation:**

Option A is valid declaration of float.

Option B is incorrect because any literal number with a decimal point u declare the computer will implicitly cast to double unless you include "F or f"

Option C is incorrect because it is a String.

Option D is incorrect because "d" tells the computer it is a double so therefore you are trying to put a double value into a float variable i.e there might be a loss of precision.

**Question 36 (ID: # 254814)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 8 |

What line of code should replace the missing statement to make this program compile?

```
/* Missing Statement ? */
public class foo
{
    public static void main(String[]args)throws Exception
    {
        java.io.PrintWriter out = new java.io.PrintWriter();
        new java.io.OutputStreamWriter(System.out,true);
        out.println("Hello");
    }
}
```

A. No statement required.
B. import java.io.*;
C. include java.io.*;
D. import java.io.PrintWriter;

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **A**

**Explanation:**

The usual method for using/importing the java packages/classes is by using an import statement at the top of your code. However it is possible to explicitly import the specific class that you want to use as you use it which is shown in the code above. The disadvantage of this however is that every time you create a new object you will have to use the class path in the case "java.io" then the class name in the long run leading to a lot more typing.

**Question 37 (ID: # 254815)**

**Subject** Core Java Level 1
**Subtopic** Java Test 2
**Description** L2 Core Java 9

What will be the output of the program?

```
public class Foo
{
    public static void main(String[] args)
    {
        try
        {
            return;
        }
        finally
        {
            System.out.println( "Finally" );
        }
    }
}
```

A. Finally
B. Compilation fails.
C. The code runs with no output.
D. An exception is thrown at runtime.

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **A**

**Explanation:**

If you put a finally block after a try and its associated catch blocks, then once execution enters the try block, the code in that finally block will definitely be executed except in the following circumstances:

1. An exception arising in the finally block itself.
2. The death of the thread.
3. The use of System.exit()
4. Turning off the power to the CPU.

I suppose the last three could be classified as VM shutdown.

**Question 38 (ID: # 254816)**

**Subject** Core Java Level 1
**Subtopic** Java Test 2
**Description** L2 Core Java 10

What will be the output of the program?

```
try
{
    int x = 0;
    int y = 5 / x;
}
catch (Exception e)
{
    System.out.println("Exception");
}
catch (ArithmeticException ae)
{
    System.out.println(" Arithmetic Exception");
}
System.out.println("finished");
```

A. finished
B. Exception
C. Compilation fails.
D. Arithmetic Exception

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

Compilation fails because ArithmeticException has already been caught.ArithmeticException is a subclass of java.lang.Exception, by time theArithmeticException has been specified it has already been caught by theException class.

If ArithmeticException appears before Exception, then the file will compile. When catching exceptions the more specific exceptions must be listed before the more general (the subclasses must be caught before the superclasses).

**Question 39 (ID: # 254817)**

**Subject** Core Java Level 1
**Subtopic** Java Test 2
**Description** L2 Core Java 11

What will be the output of the program?

```
public class X
{
    public static void main(String [] args)
    {
        try
        {
            badMethod();
            System.out.print("A");
        }
        catch (RuntimeException ex) /* Line 10 */
        {
            System.out.print("B");
        }
        catch (Exception ex1)
        {
            System.out.print("C");
        }
        finally
        {
            System.out.print("D");
        }
        System.out.print("E");
    }
    public static void badMethod()
    {
        throw new RuntimeException();
    }
}
```

A. BD
B. BCD
C. BDE
D. BCDE

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

A Run time exception is thrown and caught in the catch statement on line 10. All the code after the finally statement is run because the exception has been caught.

**Question 40 (ID: # 254818)**

**Subject** Core Java Level 1

| | |
|---|---|
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 12 |

What is the name of the method used to start a thread execution?

| | |
|---|---|
| A. | init(); |
| B. | start(); |
| C. | run(); |
| D. | resume(); |
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **B**

**Explanation:**

Option B is Correct. The start() method causes this thread to begin execution; the Java Virtual Machine calls the run method of this thread.

Option A is wrong. There is no init() method in the Thread class.

Option C is wrong. The run() method of a thread is like the main() method to an application. Starting the thread causes the object's run method to be called in that separately executing thread.

Option D is wrong. The resume() method is deprecated. It resumes a suspended thread.

**Question 41 (ID: # 254819)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 13 |

Which three are methods of the Object class?

1. notify();
2. notifyAll();
3. isInterrupted();
4. synchronized();
5. interrupt();
6. wait(long msecs);
7. sleep(long msecs);

8. yield();

| | |
|---|---|
| A. | 1, 2, 4 |
| B. | 2, 4, 5 |
| C. | 1, 2, 6 |
| D. | 2, 3, 4 |
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **C**

**Explanation:**

(1), (2), and (6) are correct. They are all related to the list of threads waiting on the specified object.

(3), (5), (7), and (8) are incorrect answers. The methods isInterrupted() andinterrupt() are instance methods of Thread.

The methods sleep() and yield() are static methods of Thread.

D is incorrect because synchronized is a keyword and the synchronized()construct is part of the Java language.

| **Question 42 (ID: # 254820)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 14 |

Which of the following line of code is suitable to start a thread?

```
class X implements Runnable
{
    public static void main(String args[])
    {
        /* Missing code? */
    }
    public void run() {}
}
```

| | |
|---|---|
| A. | Thread t = new Thread(X); |
| B. | Thread t = new Thread(X); t.start(); |
| C. | X run = new X(); Thread t = new Thread(run); t.start(); |

| | |
|---|---|
| D. | Thread t = new Thread(); x.run(); |
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **C**

**Explanation:**

Option C is suitable to start a thread.

| **Question 43 (ID: # 254821)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 15 |

Which two of the following methods are defined in class Thread?

1. start()
2. wait()
3. notify()
4. run()
5. terminate()

| | |
|---|---|
| A. | 1 and 4 |
| B. | 2 and 3 |
| C. | 3 and 4 |
| D. | 2 and 4 |
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | |

**Answer:** Option **A**

**Explanation:**

(1) and (4). Only start() and run() are defined by the Thread class.

(2) and (3) are incorrect because they are methods of the Object class. (5) is incorrect because there's no such method in any thread-related class.

**Question 44 (ID: # 254822)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 16 |

Which of the following will directly stop the execution of a Thread?

| | |
|---|---|
| A. | wait() |
| B. | notify() |
| C. | notifyall() |
| D. | exits synchronized code |

| | |
|---|---|
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **A**

**Explanation:**

Option A is correct. wait() causes the current thread to wait until another thread invokes the notify() method or the notifyAll() method for this object.

Option B is wrong. notify() - wakes up a single thread that is waiting on this object's monitor.

Option C is wrong. notifyAll() - wakes up all threads that are waiting on this object's monitor.

Option D is wrong. Typically, releasing a lock means the thread holding the lock (in other words, the thread currently in the synchronized method) exits the synchronized method. At that point, the lock is free until some other thread enters a synchronized method on that object. Does entering/exiting synchronized code mean that the thread execution stops? Not necessarily because the thread can still run code that is not synchronized. I think the word directly in the question gives us a clue. Exiting synchronized code does not directly stop the execution of a thread.

**Question 45 (ID: # 254823)**

| | |
|---|---|
| **Subject** | Core Java Level 1 |

| | |
|---|---|
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 17 |

Which method must be defined by a class implementing the java.lang.Runnableinterface?

| | |
|---|---|
| A. | void run() |
| B. | public void run() |
| C. | public void start() |
| D. | void run(int priority) |

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation

**Answer:** Option **B**

**Explanation:**

Option B is correct because in an interface all methods are abstract by default therefore they must be overridden by the implementing class. The Runnable interface only contains 1 method, the void run() method therefore it must be implemented.

Option A and D are incorrect because they are narrowing the access privileges i.e. package(default) access is narrower than public access.

Option C is not method in the Runnable interface therefore it is incorrect.

| **Question 46 (ID: # 254824)** | |
|---|---|
| **Subject** | Core Java Level 1 |
| **Subtopic** | Java Test 2 |
| **Description** | L2 Core Java 18 |

which of these will create and start this thread?

```
public class MyRunnable implements Runnable
{
    public void run()
    {
        // some code here
    }
}
```

A. new Runnable(MyRunnable).start();
B. new Thread(MyRunnable).run();
C. new Thread(new MyRunnable()).start();
D. new MyRunnable().start();

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation

**Answer:** Option **C**

**Explanation:**

Because the class implements Runnable, an instance of it has to be passed to theThread constructor, and then the instance of the Thread has to be started.

A is incorrect. There is no constructor like this for Runnable because Runnable is an interface, and it is illegal to pass a class or interface name to any constructor.

B is incorrect for the same reason; you can't pass a class or interface name to any constructor.

D is incorrect because MyRunnable doesn't have a start() method, and the onlystart() method that can start a thread of execution is the start() in the Threadclass.

**Question 47 (ID: # 152249)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Collections |
| **Description** | Descending List of Names |

Which of the following options, inserted in the code below, will cause it to compile and output a descending list of names?

```
List<String> names = new ArrayList<String>();
        names.add("James");
        names.add("Hanna");
        names.add("Walter");
        names.add("Robert");
        names.add("Manny");

Collections.sort(names, new Comparator<String>() {
        public int compare(String s1, String s2) {
                // insert code here
        }
});

        for (String name : names)
                System.out.println(": " + name);
```

A. return s1.contentEquals(s2);
B. return -1 * s1.equalsIgnoreCase(s2);
C. return s1.equalsIgnoreCase(s2);
D. return s2.compareToIgnoreCase(s1);
E. return s1.compareToIgnoreCase(s2);

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation The Comparator interface allows you to sort a given collection any number of different ways. You can use it to sort instances of any class, even classes you can't alter. This is unlike the Comparable interface, which requires you to alter the class whose instances you wish to sort.
The compare method returns an int value. Thus, equalsIgnoreCase and contentEquals are not suitable because they return Boolean values.
If you want the output list to be in descending order then change the argument to compareIgnoreCase so that the first string is passed.

**Question 48 (ID: # 152267)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Autoboxing/Unboxing
**Description** Override

What will the output be when the program below is run?

```
class Bebe {
        void play(Bebe a) { System.out.print(" bebe"); }
}

class Child extends Bebe {
        public static void main(String[] args) {
                Child c = new Child();
                Bebe b = new Bebe();
                Bebe b1 = new Child();

                c.play(c);
                b.play(b);
                b.play(b1); // 1
                b1.play(b); // 2
        }

void play(Bebe a) { System.out.print(" child"); }
}
```

A. child bebe child bebe
B. child bebe bebe child
C. Runtime error at line 1.
D. Compiler at line 1.
E. Compiler error at line 2.

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation Child.play(Bebe a) overrides Bebe.play(Bebe a) .
The compiler widens the Child reference to a Bebe , and the play() method invocation succeeds.
The reference widening depends on the inheritance.

**Question 49 (ID: # 152268)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Autoboxing/Unboxing
**Description** Errors

Which lines in the code below will bring out compiler errors?

```
class Test {
        public static void main(String[] args) {
                int a = 3;
                Byte b = 5;
                long c = 2;

                compute(a); // 1
                compute(new Long(4)); // 2
                compute(b); // 3
                compute(c); // 4
                compute(new Integer(4)); // 5
        }
        static void compute(Long x) { System.out.print(x); }
}
```

A. Lines 1 and 3
B. Lines 3 and 5
C. Lines 1, 3, and 5
D. Line 5
E. Line 3

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The wrapper classes do not widen from one to another. The reference widening depends on the inheritance. Thus wrapper classes do not inherit from one another.
Line 1: a is boxed to Integer and Integer does not widen to Long .
Line 2: There is no need for autoboxing, and no compiler error occurs.
Line 3: Byte does not widen to Long .
Line 4: long is boxed to Long , so no compiler error occurs.
Line 5: Integer does not widen to Long .

**Question 50 (ID: # 152269)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Autoboxing/Unboxing
**Description** Variables

What will the output be when the program below is run?

```
class Test {
        static Map<String, Object> map = new LinkedHashMap<String, Object>();
        static void put(long x) { map.put("long", x); }
        static void put(Float x) { map.put("Float", x); }
        static void put(Integer x) { map.put("Integer", x); }
        static void put(int... x) { map.put("int...", x); }

        public static void main(String[] args)
        {
                int a = 3;
                Double b = 3D;
                put(b.floatValue()); // 1
                put(a);
                for(Iterator<String> it = map.keySet().iterator(); it.hasNext();) {
                        System.out.print(" ", it.next());
                }
        }
}
```

A. Float long
B. Float int...
C. float Integer
D. float
E. Compiler error at line 1.

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation The double value of the b variable is converted to float .
The compiler will always choose widening over boxing or var-args , but in this case it has no widening alternative, and boxing comes before var-args , so boxing is chosen. The put() call boxes the float to a Float .
The variable a is an int . In this case, the following occurs.
- put(long x) offers the widening alternative.
- put(Integer x) offers the boxing alternative.
- put(int... x) offers the widening alternative.

Following the logic above, widening wins in this case.

**Question 51 (ID: # 152272)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Autoboxing/Unboxing
**Description** Variable Types

For the code below, what are the possible types of variable `z`?

```
byte x = 7;
short y = 3;
z = x * ++y;
```

A. short, int, long, float, double
B. int, long, float, double
C. short, char, int, float, double
D. byte, short, int, long, float, double
E. short, char, int, float, double, string

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation A literal integer is always an int , and the result of an expression involving anything int -sized or smaller is always an int .
In other words, the expression x * ++y is evaluated to an int , and the z variable can be declared as int and all of the other bigger types.
The primitive types are char(16), boolean(true/false), byte(8), short(16), int(32), long(64), double(64) and float( 32) .

**Question 52 (ID: # 152252)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Collections
**Description** Collections Class

What will the output be when the code below is run?

```
public class HotelManagement {
	public static void main(String[] args) {

		List<Hotel> hotels = new ArrayList<Hotel>();

		hotels.add(new Hotel("Meriot", 4)); // 1
		hotels.add(new Hotel("Ritz", 5));
		hotels.add (new Hotel("Hilton", 3));

		Collections.sort(hotels); // 2
		for (Hotel hotel : hotels) {
			System.out.println(hotel.name + hotel.stars); // 3
		}
	}
}
class Hotel {
	String name = "hotel";
	protected Integer stars;

	Hotel(String name, int stars) {
		this.name = name;
		this.stars = stars;
	}
}
```

A. Compiler error at line 1.
B. Runtime error at line 1.
C. Compiler error at line 2.
D. Runtime error at line 2.
E. Compiler error at line 3.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The Hotel class does not implement the Comparable interface. Therefore, you cannot sort any collection without giving it a proper Comparable implementation. The Comparable interface is used by the Collections. sort() method and the java.util.Arrays.sort() method to sort lists and arrays of objects, respectively. To implement Comparable , a class must implement a single method, compareTo() .

**Question 53 (ID: # 152253)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Collections
**Description** Map

What will the output be when the code below is run?

```
TreeMap<String, String> map = new TreeMap<String, String>();
map.put("A", "Audi"); map.put("D", "Dodge"); map.put("L", "Lamborghini");
SortedMap<String, String> submap;
submap = map.subMap("B", "K");
map.put("B", "BMW");
map.remove("A");
submap.put("F", "Ferrari");
map.put("R", "Rover");
submap.put("J", "Jaguar");
submap.put("P", "Porsche");
submap.remove("F");
System.out.println(map + " " + submap);
```

A. Compile error.
B. {B=BMW, D=Dodge, J=Jaguar, L=Lamborghini, R=Rover} {B=BMW, D=Dodge, J=Jaguar}
C. {B=BMW, D=Dodge, L=Lamborghini, P=Porsche, R=Rover} {B=BMW, D=Dodge}
D. {B=BMW, D=Dodge, L=Lamborghini, R=Rover} {B=BMW, D=Dodge}
E. Runtime error.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The subMap() method makes a copy of a portion of the TreeMap named map. When the submap was created, a value range was provided for the new collection. This range defines not only what should be included when the partial copy is created, but also the range of values that can be added to the copy.
In this case, an attempt is made to put ("P", "Porsche") to the submap, but this outruns the range of the submap.

**Question 54 (ID: # 152254)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Collections
**Description** List

What will the output be when the following program is run?

```
List<Integer> list = new ArrayList<Integer>();
list.add(new Integer(1));
list.add(new Integer(2));
list.add(new Integer(3));
Iterator<Integer> iterator = list.iterator();
while(iterator.hasNext()) {
        System.out.print(" " + iterator.next());
        for (Iterator<Integer> i = iterator; iterator.hasNext(); iterator.next()) {
                System.out.print(" " + 2 * i.next());
        }
}
```

A. 1 4 6

| | |
|---|---|
| B. | 1 2 3 |
| C. | 1 2 4 |
| D. | 1 4 |
| E. | 1 4 3 |
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The two Iterator methods involved are the following.<br>- boolean hasNext() returns true if there is at least one more element in the collection being traversed. Invoking hasNext() does not move you to the next element of the collection.<br>- Object next() returns the next object in the collection, and moves you forward to the element after the element just returned.<br>Therefore, at Line 7, iterator.next() returns the first element in the list, since the default cursor is 0 (and it is increased after that).<br>At Line 8, iterator.hasNext() returns true because the list contains the Integer(2) value at Index 1. The iterator. next() is not invoked until the end of the for block, so the cursor remains at 1 .<br>At Line 9, i.next() returns the Integer value from Position 1 in the list, and right after that it increases the cursor to 2 , so the output becomes 2 * i.next() = 2 * 2 = 4 .<br>At Line 10, the iterator.next() is invoked as the end of the for block is reached. But the list contains only three elements, so the cursor cannot be increased further (it takes values from 0, 1, 2 ).<br>In Line 8, the iterator.hasNext() returns false and exits the for block.<br>As at Line 6, the iterator.hasNext() returns false and exits the while block. |

**Question 55 (ID: # 152257)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Collections |
| **Description** | Compile |

What will the output be when the code below is run?

```
class CarDealer {
        public static void main(String[] args) {
                ArrayList<Model> models = new ArrayList<Model>();
                models.add(new Model("Audi", 10));
                models.add(new Model("BMW", 20));
                models.add(new Model("Mercedes Benz", 5));
                Collections.sort(models, new Model("Others", 50)); // 1
                for(Model model : models)
                        System.out.print(model.brand + " ");
        }
}
class Model implements Comparator<Model> {
        String brand;
        int count;
        Model(String s, int n) {
                this.brand = s;
                this.count = n;
        }
        public int compareTo(Model o) {
                return this.count -  o.count;
        }
        public int compare(Model o1, Model o2) {
                return o2.count - o1.count ;
        }
}
```

A. Compile error at line 7.

B. BMW Audi Mercedes Benz

C. Mercedes Benz Audi BMW Others

D. Mercedes Benz Audi BMW

E. BMW Audi Mercedes Benz Others

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation The Model class implements the Comparator interface, thus the compare(...) method is used in the Comparator implementation for sorting.
The new Model("Others", 50) is the actual Comparator implementation, and it's not added to the sort process among the other models. Therefore, Options A, C, and E are incorrect.
The model list is in descending order, given that you subtract ((Model)o1).count from ((Model)o2).count , so switching the place of the operands reverses the natural order. Therefore, Option D is incorrect.

**Question 56 (ID: # 152258)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Collections

**Description** HashMap

Which of the following options, when inserted as indicated in the code below, will make the code compile?

```
public class PotionStore {
        public static void main(String[] args) {
                // insert code here
                potions.put("strength", "make a person gain superhuman powers");
                potions.put("heal", "make a person to heal itself");
                potions.put("sleeping", "cause a person to fall asleep");
                System.out.println(potions.get("heal"));
        }
}
```

A. HashMap potions = new LinkedHashSet();

B. Map potions = new HashMap();

C. HashMap potions = new TreeMap();

D. Map potions = new TreeMap();

E. Hashtable potions = new HashMap();

| | |
|---|---|
| Correct Answer | B ;D ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Options B and D are correct. Option A is incorrect because the types are incompatible: HashMap implements Map , while LinkedHashSet implements HashSet . Option C is incorrect because although HashMap and TreeMap are both maps, they have different implementations. |

**Question 57 (ID: # 152259)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Collections |
| **Description** | Ordered Lists |

Which of the following options will make the code below compile and print an ordered list of motels?

```
class Motel {
        String name; int stars;
        Motel(String n, int s) { name = n; stars = s; }
}
public class Agency {
        List <Motel> ml = new ArrayList <Motel> ();
        class MotelComp /* fragment 1 */ {
        public int /* fragment 2 */(Motel one, Motel two) {
                return one.stars - two.stars;
        }
}
public static void main(String[] args) {
        new Agency().listMotels();
}
void listMotels() {
        ml.add(new Motel("Cozzy Cone", 3));
        ml.add(new Motel("Shamrock Inn", 4));
        ml.add(new Motel("Little Texas", 2));

        MotelComp c = new MotelComp();
        Collections./* fragment 3 */
        for( Motel m : ml)
                System.out.println(ml.indexOf(m) + 1 + ". " + m.stars + " " + m.name);
        }
}
```

A. fragment 1 : implements Comparable <Motel>

B. fragment 1 : implements Comparator <Motel>

C. fragment 2 : compareTo

D. fragment 2 : compare

E. fragment 3 : sort(ml, c);

Correct Answer B ;D ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation The sorting is done using a Comparator interface.
The differences between the Comparator and Comparable interfaces are listed below.
- int compare(objOne, objTwo)(Comparator) / int objOne.compareTo(objTwo)(Comparable)
- Both of these return the same results:
negative if objOne < objTwo ;
zero if objOne == objTwo ; and
positive if objOne > objTwo .

- For the Comparator interface, you can build a class separate from the class whose instances you want to sort. For the Comparable interface, you must modify the class whose instances you want to sort.
- For the Comparator interface, many sort sequences can be created. For the Comparable interface, only one sort sequence can be created.
The sortList() function does not exist.

**Question 58 (ID: # 17080)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Exceptions Handling |
| **Description** | Method not Allowed to Throw in Finally |

Given the code below, and assuming that all `FileOutputStream` methods throw an `IOException`, which of the following statements is true?

```
1. import java.io.*;
2. public class MyLogger {
3.         public static void main(String[] args) {
4.                 FileOutputStream out = null;
5.                 try {
6.                         out = new FileOutputStream("numbers.bin");
7.                         out.write(17);
8.                 } catch (IOException ioe) {
9.                         System.out.println("I/O exception");
10.                } finally {
11.                        out.close();
12.                }
13.        }
14. }
```

A. The program will compile successfully.

B. The program will fail to compile due to an error on line 6.

C. The program will fail to compile due to an error on line 7.

D. The program will fail to compile due to an error on line 8.

E. The program will fail to compile due to an error on line 11.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The out.close() throws an IOException as well, and it must be caught in the finally statement and in the main method.

**Question 59 (ID: # 17081)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Exceptions Handling |
| **Description** | Finally Statement Properties |

Which of the following statements is true regarding the `finally` clause of a `try-catch-finally` block?

A. It doesn't execute unless one catch block didn't execute.

B. It cannot hold a nested try-catch block inside it.

C. It is not executed if no exception was thrown.

D. It is always executed if its thread is not finished.

E. It cannot throw the same exception because this will be garbage collected after finally finishes.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation The purpose of the finally block is to make sure some code is executed, no matter what exception was thrown. Therefore, only Option D is correct.

**Question 60 (ID: # 17082)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Exceptions Handling

**Description** Catch Order

What will the result be when the code below is executed?

```
1. public class Test {
2.    public static void main(String[] args) {
3.           try {
4.                throw new MyException2();
5.           } catch(MyException1 me1) {
6.                System.out.println("MyEx1");
7.           } catch(MyException2 me2) {
8.                System.out.println("MyEx2");
9.           }
10.
11.   }
12. }
13.
14. class MyException1 extends Exception {}
15. class MyException2 extends MyException1 {}
```

A. It will print "MyEx2".

B. It will print "MyEx1".

C. It will print "MyEx1 MyEx2".

D. It will fail to compile at line 5.

E. It will fail to compile at line 7.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation Compilation of this code will fail at line 7. The compiler will state that the exception MyException2 is already caught (by catching MyException1 , since MyException2 is a subclass of MyException1 ).

**Question 61 (ID: # 17083)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Exceptions Handling

**Description** Throw Exception in Catch Block

What will the result be when the code below is run?

```
1. public class Test {
2.    public static void main(String[] args) {
3.           try {
4.                throw new MyException1();
5.           } catch(MyException1 me1) {
6.                System.out.println("MyEx1");
7.                throw new MyException2 ();
8.           } catch(MyException2 me2) {
9.                System.out.println("MyEx2");
10.          }
11.
12.   }
13. }
14.
15. class MyException1 extends Error {}
16. class MyException2 extends Error {}
```

A. It will fail to compile at Line 7.

B. It will fail to compile at Line 8.

C. It will print "MyEx1 MyEx2".

D. It will print "MyEx1" then a stack trace.

E. It will print a stack trace.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation Although MyException1 and MyException2 are subclasses of Error , they can still be caught, and this is the case of MyException1 . However, the new MyException2 that is thrown is not caught immediately in the next ( otherwise matching) catch statement. But the compiler will not complain that it is not caught because MyException2 is an unchecked exception. So this exception is not caught and ends up filling the console with a stacktrace. Therefore, Option D is correct.

**Question 62 (ID: # 17084)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Exceptions Handling

**Description** Unchecked Exceptions

Which of the following Exceptions do not need to be caught or specified in the method declarations?

A. exceptions that subclass Error

B. exceptions that subclass Throwable

C. exceptions that subclass RuntimeException

D. exceptions that subclass Exception

E. All exceptions must be caught or declared.

Correct Answer A ;C ;

User Answer

Elapsed Time (seconds) 0

Explanation Errors and RuntimeExceptions are the so-called unchecked exceptions, which do not need to be caught or declared to be thrown. These exceptions can be caught, but usually doing so doesn't help solve a logic error,

since when they are thrown the system or the application is in an unrecoverable state. All of the exceptions that are not instanceof RuntimeExceptions or Errors must be caught or declared.

**Question 63 (ID: # 17085)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Exceptions Handling |
| **Description** | Exception Declarations in Method Signatures |

Given the code below, which exceptions may be thrown from inside the method, so that the compiler won't have any difficulties?

```
public void throwsAnException()
throws MyException1, MyException2  {
      // some code
}
```

A. MyException1
B. MyException2
C. subclasses of MyException1
D. subclasses of Error
E. subclasses of Throwable

Correct Answer A ;B ;C ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation MyException1 , MyException2 , and all of their subclasses may be thrown from inside the method. Throwable is the root class for Errors and Exceptions . Therefore, we can't know that all the subclasses can be thrown. However, subclasses of Errors and RuntimeExceptions (which are Throwable ) may be thrown without being declared in the method's signature. Options A, B, C, and D are correct.

**Question 64 (ID: # 17086)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Exceptions Handling |
| **Description** | Overriding Method Exception |

An overriding method may throw more exceptions than the overridden one, but it must declare them explicitly.

Correct Answer FALSE

User Answer

Elapsed Time (seconds) 0

Explanation An overriding method may throw at most the exceptions declared by the overridden method. An overriding method may also throw any unchecked exceptions (as well as the overridden exceptions).

**Question 65 (ID: # 17087)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |

| | |
|---|---|
| **Subtopic** | Exceptions Handling |
| **Description** | Checked Exceptions |

A(n) _____ exception may be thrown, but it must either be caught in a try-catch block or it must be declared, in order to be thrown in the enclosing method.

A. Runtime
B. Checked
C. Unchecked
D. Application
E. System

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation Exceptions fall into two categories: those that must be caught (called "checked exceptions"), and those that may not be caught explicitly (called "unchecked exceptions"). Unchecked exceptions include Runtime exceptions and errors.

**Question 66 (ID: # 17088)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Exceptions Handling |
| **Description** | Exceptions Handling in Method Overriding |

What will the result be when the code below is run?

```
import java.io.*;

class Base {
      public static void amethod() throws FileNotFoundException {}
}

public class ExceptionDemo extends Base {
      public static void main(String argv[ ] ) {
            ExceptionDemo e = new ExceptionDemo();
      }
      public static void amethod() {}
      protected ExceptionDemo() {
            try {
                  DataInputStream din = new DataInputStream(System.in);
                  System.out.println("Pausing");
                  din.readChar();
                  System.out.println("Continuing");
                  this.amethod();
            } catch (IOException ioe) {}
      }
}
```

A. A compile time error will occur, caused by the protected constructor.
B. A compile time error will occur, caused by amethod not declaring an exception.
C. A runtime error will occur, caused by the amethod not declaring an exception.
D. The code will compile and run with output.
E. The code will compile and run without output.

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | An overridden method in a subclass should not throw exceptions that are not thrown in the base class. The method amethod() , however, throws no exceptions and will therefore compile without complaint. There is no basis to conclude that a constructor cannot be protected. |

**Question 67 (ID: # 152242)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Generics |
| **Description** | Lists |

The code fragment below will print "true."

```
List <String> l1 = new ArrayList<String>();
List<Integer> l2 = new ArrayList<Integer>();
System.out.println(l1.getClass() == l2.getClass());
```

| | |
|---|---|
| Correct Answer | TRUE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | You may be inclined to think that it will print "false," but that would be incorrect. It will print "true," because all instances of a generic class own the same run-time class, despite their actual type parameters. |

**Question 68 (ID: # 152243)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Generics |
| **Description** | Wildcards |

The code below will not compile because it is attempting to pass the wrong type to the method `doIt`. Which of the following wildcards should be used instead of the `object` type inside the `doIt` method?

```
package com.brainysoftware.jdk5.app16;
import java.util.ArrayList;
import java.util.List;

public class AllowedTypeTest {
	public static void doIt(List<Object> l) {
	}

	public static void main(String[] args) {
		List<String> myList = new ArrayList<String>();
		// this will generate a compile error
		doIt(myList);
	}
}
```

| | |
|---|---|
| A. | ? extends Type |
| B. | ? super Type |
| C. | ? |
| D. | * |
| E. | * super Type |
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | This code won't compile, because it is attempting to pass the wrong type to the method doIt . doIt waits for an instance of Listand instead you are passing it an instance of List . The resolution to this issue is the ? wildcard. List represents a list of objects of any type. Consequently, the doIt method should be modified to the following.<br><br>public static void doIt(List l) {<br>} |

**Question 69 (ID: # 152244)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Generics |
| **Description** | Working with Raw Types |

In J2SE 5.0, which of the following options can you use if you do not want to get warnings when working with raw types?

| | |
|---|---|
| A. | Compile with the -source 1.4 flag . |
| B. | Use the @SupressWarnings("unchecked") annotation. |
| C. | Infer type arguments for references to generic types |
| D. | Use the @Override("unchecked") annotation. |
| E. | Compile with the -source 1.3 flag . |
| Correct Answer | A ;B ;C ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Options A, B, and C are correct. Instances of Listmay accept any type of object and act like a raw type List . Yet, the compiler will not complain. A generic type employed with no parameters is named a raw type. This signifies that code written for JDK 1.4 and previous versions will continue to work in Java 5. |

**Question 70 (ID: # 152247)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Generics |
| **Description** | List in a List |

Given the code below, which of the following syntaxes would you use to retrieve the first `String` from the first `List` in `myList`?

```
List<List<String>> myListOfListsOfStrings;
```

A. String s = myListOfListsOfStrings.get(0).get(0);
B. String s = myListOfListsOfStrings.get(0);
C. String s = myListOfListsOfStrings.get(1).get(1);
D. String s = myListOfListsOfStrings.get(1);
E. String s = myListOfListsOfStrings.get(0).get(1);

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation A generic type is itself a type and can be used as a type variable. For instance, if you wish for your List to store lists of Strings , you may declare the List by passing List as its type variable.

**Question 71 (ID: # 152248)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Generics
**Description** java.lang.Throwable

A generic type must not be a direct or indirect child class of _____.

A. java.lang.ThreadLocal
B. java.lang.Void
C. java.lang.Throwable
D. java.lang.System
E. java.lang.String

Correct Answer C ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation A generic type should not be a direct or indirect child class of java.lang.Throwable , because exceptions are thrown at Run time. For that reason, it is impossible to predict what type of exception may be thrown at Compile time. Also it should not inherit java.lang.String because it is declared as final .

**Question 72 (ID: # 17089)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Environment Features
**Description** Hash and Equals

Given the code below, and assuming it prints "r = 1111" which of the following statements are definitely true?

```
int r = 0;

if (o1.hashCode() != o2.hashCode()) r+=1;
if (o3.equals(o4)==false) r+=10;
if (o5.equals(o6)) r+=100;
if (o7.hashCode() == o8.hashCode()) r+=1000;

System.out.println("r = "+r);
```

A. o8.equals(o7) == true

B. o2.equals(o1) == true

C. o3.hashCode() != o4.hashCode()

D. o5.hashCode() == o6.hashCode()

E. o7.hashCode(o8) != o4.hashCode()

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation By contract, if two objects are equivalent according to the equals() method, then their hashCode() method must evaluate them to be == . Option A is incorrect because hashCode() will often return == , even if the two compared objects do not return true on equals() . Option B is incorrect because if the hashCode() values are not equal, the two objects must not be equal. Option C is not correct because if equals() is not true , then no assumption can be made about the hashCode() . Option D is correct because the objects are equivalent according to the equals() , so their hashCode() must be == according to the contract.

**Question 73 (ID: # 17090)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Java Environment Features

**Description** Object Reference Reachability

Which of the following are levels of reachability of an object?

A. Strong

B. Phantom

C. Loose

D. Pseudo

E. Chained

Correct Answer A ;B ;

User Answer

Elapsed Time (seconds) 0

Explanation Going from strongest to weakest, the different levels of reachability reflect the life cycle of an object. They are operationally defined as follows.
- An object is strongly reachable if it can be reached by a thread without traversing any reference objects. A newly-created object is strongly reachable by the thread that created it.
- An object is softly reachable if it is not strongly reachable but can be reached by traversing a soft reference.
- An object is weakly reachable if it is neither strongly nor softly reachable, but can be reached by traversing a weak reference. When the weak references to a weakly-reachable object are cleared, the object becomes eligible for finalization.

- An object is phantom reachable if it is neither strongly, softly, nor weakly reachable, it has been finalized, and some phantom reference refers to it.
- Finally, an object is unreachable, and therefore eligible for reclamation, when it is not reachable in any of the above ways.

**Question 74 (ID: # 17091)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | One-Hash-Fits-All |

The method below is a legal method for a `hashCode` implementation.

```
public int hashCode() {
      return 1723;
}
```

| | |
|---|---|
| Correct Answer | TRUE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Even though this method is terribly inefficient for a search, it is a legal and correct method for providing an object's hash code, because it obeys all of the rules in the contract: reflexivity, symmetry, transitivity, and consistency. |

**Question 75 (ID: # 17092)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | hashCode() efficiency |

Which of the following statements are true for the code below?

```
1. class Test1 {
2.       public int value;
3.       public int hashCode() { return 17; }
4.       }
5. class Test2 extends Test1 {
6.       public int hashCode() throws ArithmeticException
7.       { return 1000 / value;}
8. }
```

A. The compiler will complain about line 6.

B. The compiler will complain about line 7.

C. The hashing method of class Test1 is more effective than the method of Test2 .

D. The hashing method of class Test2 is legal.

E. The two hashCode methods have the same efficiency.

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation For any non-null reference values x and y, several invocations of x.equals(y) consistently return true or consistently return false , provided that no information used in equals comparisons on the objects is changed.

**Question 76 (ID: # 17093)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Environment Features
**Description** Classes Inheriting HashCode and Equals

Which of the following classes inherits and uses the default implementations of `hashCode()` and `equals()` methods directly from `java.lang.Object`?

A. java.lang.System
B. java.lang.String
C. java.lang.Double
D. java.lang.Math
E. java.lang.Character

Correct Answer A ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation Options A and D are correct. Among the methods inherited from class java.lang.Object are the hashCode() and equals() methods.

**Question 77 (ID: # 17094)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Environment Features
**Description** Available for GC

How many objects are eligible for garbage collection just before the code on Line 9 gets executed?

```
1. public class G
2. {
3.       public static void main(String [] args)
4.       {
5.             G g1 = new G();
6.             G g2 = alloc(g1);
7.             G g3 = new G();
8.             g2 = g1;
9.             System.out.println("Hello world!");
10.      }
11.      static G alloc (G g)
12.      {
13.            g = new G();
14.            return g;
15.      }
16. }
```

A. 0
B. 1
C. 2

| | |
|---|---|
| D. | 3 |
| E. | 4 |
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The only object without a reference is the one generated as a result of the call in Line 5. Java passes a copy of the reference variable as a parameter so the variable g1 is not affected by the allocations within the method. |

**Question 78 (ID: # 17095)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | Garbage Collection Behavior |

Which of the following statements are true regarding garbage collection?

| | |
|---|---|
| A. | Objects with the finalize() method overridden will never be garbage collected. |
| B. | Objects instantiated within anonymous inner classes are placed in the garbage collectable heap. |
| C. | All objects that are eligible for garbage collection will be garbage collected at the next sweep. |
| D. | Objects with at least one reference will never be garbage collected. |
| E. | Once the finalize() method is invoked, there is no way to make that object ineligible for garbage collection. |
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Option B is correct, because all of the objects created will be placed on the heap and be checked for collecting by the garbage collector. |

**Question 79 (ID: # 17096)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | Garbage Collector Behavior |

Which of the following statements is true?

| | |
|---|---|
| A. | Calling Runtime.gc() will cause eligible objects to be garbage collected. |
| B. | If an object can be accessed from a live thread, it can't be garbage collected. |
| C. | If Object 1 refers to Object 2 and Object 2 refers to Object 1, then they cannot be garbage collected. |
| D. | The garbage collector uses a "mark and sweep" algorithm for collecting. |
| E. | When an OutOfMemoryError is caught, Runtime.gc() will help free up some memory. |
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation The basic assumption for the garbage collector is that if an object is referred from a live context it is not eligible for collecting. Thus, Option B is correct. Options A and D make assumptions that the garbage collector doesn't make. Option C represents a case of "islands of isolated objects" that can be detected by the collector and swept away. Usually an error is not caught because it is futile, since the application is already irrecoverably corrupted. Moreover, running the collector (if possible) doesn't guarantee anything.

**Question 80 (ID: # 17097)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Environment Features
**Description** Eligible for Garbage Collecting

Given the code below, which of the following pieces of code may be inserted at the specified location, in order to obtain an object eligible for garbage collecting?

```
X3 x2 = new X3();
X3 x3 = new X3();
X3 x5 = x3;
x3 = x2;
X3 x4 = x3;
x2 = null;
//insert code here
```

A. x3 = x4;
B. x3 = null;
C. x5 = null;
D. x5 = x4;
E. x4 = null;

Correct Answer C ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation By the time the insertion line is reached, there are two objects on heap: one referred by x3 and x4 , and the other referred by x5 . Therefore, cutting the last reference from x5 solves the problem.

**Question 81 (ID: # 17098)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Environment Features
**Description** NewInstance Exceptions

Which of the following exceptions must be declared for the method below, in order for it to compile successfully?

```
public MyClass createMyInstance(Class clazz)
/* throws what exceptions? */
{
      Object o = clazz.newInstance();
      MyClass myInstance = null;

      try {
            myInstance = (MyClass)o;
      } catch (ClassCastException cce) {
            System.out.println("Illegal class: "+clazz.getName());
      }

      return myInstance;
}
```

A. SecurityException
B. InstantiationException
C. ExceptionInInitializerError
D. NoSuchMethodException
E. IllegalAccessException

Correct Answer B ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation The newInstance() method throws exceptions from Options A, B, C, and E, but only InstantiationException and IllegalAccesException need to be declared, since the other two are unchecked exceptions.

**Question 82 (ID: # 17099)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Environment Features
**Description** getMethod behavior

What is the expected behavior of the `Class.getMethod` method, when a method name of "<init>" is passed to it?

A. A null is returned.
B. A method representing a constructor is returned.
C. An IllegalAccessException is thrown.
D. An NoSuchMethodException is thrown.
E. An AccessException is thrown.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation If the name is ""or "," a NoSuchMethodException is raised. Otherwise, the reflected method is searched in the class, in the superclasses, and in the superinterfaces.

**Question 83 (ID: # 17100)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | Superclass of Field and Method |

Which of the following is the common superclass of classes `Field`, `Method`, `Constructor`, and `Modifier`?

A. java.lang.reflect.AccessibleObject
B. java.lang.reflect.AbstractObject
C. java.lang.reflect.ReflectionElement
D. java.lang.reflect.ReflectionObject
E. none of above

| | |
|---|---|
| Correct Answer | E |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The four classes do not have any common superclass. The first three extend from a common class, AccessibleObject , but Modifier extends directly from Object . |

**Question 84 (ID: # 17101)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | Two Interfaces in a File |

Two interfaces can be declared in a single source file, if you specify the proper access modifiers.

| | |
|---|---|
| Correct Answer | TRUE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | In a source file, any number of classes and/or interfaces can be declared. The only restriction (apart from any name conflicts) is that there can be at most one public interface or class, and its name needs to match the file's name. |

**Question 85 (ID: # 17102)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | Overloading Main() |

The code below will compile. (The import statements are okay.)

```
public class Test {
      public static final void main(String[] arguments)
      throws ClassNotFoundException {
            Class.forName("this.clazz.does.not.Exist");
      }
}
```

| | |
|---|---|
| Correct Answer | TRUE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The code will compile, because the main function has not been overloaded. Its overriding capability has been modified, but it will compile just fine. However, if you try to run it, a ClassNotFoundException will be thrown. |

**Question 86 (ID: # 17103)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Environment Features |
| **Description** | Inheriting Main() |

What will happen if you try to compile and run class `Test`, shown below?

```
public class Test extends BaseTest {}

class BaseTest {
      public static void main(String[] arguments) {
            System.out.println("main!");
      }
}
```

A. It will print "main!".
B. The compiler will complain about inheriting the special function main() .
C. The compiler will complain about inheriting a static method.
D. The compiler will complain about missing main in public class.
E. It will compile but it will throw a NoSuchMethodException .

| | |
|---|---|
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The method main is not a special method for an object. It may be inherited like any other method. So the code will compile and print the text "main!". |

**Question 87 (ID: # 17047)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Capacity Overflow |

What will the result be when the code below is run?

```
int n = 2000000000;
System.out.println(n * n);
```

A. 0
B. NaN
C. -1651507200
D. An ArithmeticException will be thrown.
E. 4000000000000000000

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation 2 000 000 000 is represented as a binary 01110111 00110101 10010100 00000000 . 2 000 000 000 * 2 000 000 000 = 4 000 000 000 000 000 000 , which is internally represented as 00110111 10000010 11011010 11001110 10011101 10010000 00000000 00000000 . But an int has only 32 digits, so it can't hold this value. However, no exception is thrown. Instead, the number is truncated to the least significant 32 digits, which yields a negative number. Thus, the piece of code will run and output -1651507200 (because of the bit of sign, a negative number occurs).

**Question 88 (ID: # 17048)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Language
**Description** Increment and Decrement

What will the output be when the code below is run?

```
int m = 100;
int n = 300;

while(++m < n--);
System.out.print(m);
```

A. 199
B. 200
C. 201
D. 300
E. 301

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation In the above code fragment, the value of m is 100 and the value of n is 300. The While loop will continue until the value of n becomes less than the value of m. However, the variable n is incremented after the expression is evaluated, so the result will be 201.

**Question 89 (ID: # 17049)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Infinite Loop |

What will the result be when the code below is run?

```
byte b = 1;

while(b++ > 0) {
      b&=b;
}

System.out.print("Hi there!");
```

A. an infinite loop
B. The program will print "Hi there!".
C. A compilation error will be generated at compile time.
D. The program will print "Hi there!" two times.
E. The program will print "Hi there!" five times.

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The variable 'b' will go up to 127. The logic operator AND is idempotent, meaning that b & b == b for any value of b, so the line inside the loop doesn't affect the flow. After the overflow occurs, 'b' will be set to a negative value. Thus, the loop will end, and it will print "Hi there!". |

**Question 90 (ID: # 17050)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | OR Operator |

What will the result be when the code fragment below is run?

```
System.out.println(2 | 3);
```

A. 6
B. 5
C. 0
D. 3
E. An error will occur.

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The bitwise operator (\|) is the OR operator in Java, which turns the given number into its binary equivalent and then performs a Boolean/OR operation. The binary equivalent of 2 is "010," and of 3 is "011". For each position |

, if either number contains a 1 the result will contain a 1 in that position. The result will be 011. The decimal equivalent is 3.

**Question 91 (ID: # 17051)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Break Statement in Switch Block |

What will the output be when the code below is run?

```
int i = 9;

switch(i){
      default :
            System.out.print("default ");
      case 0 :
            System.out.print("zero ");
      case 1 :
            System.out.print("one ");
      case 2 :
            System.out.print("two ");
}
```

A. default
B. default zero
C. error - default value not defined
D. No output will be displayed.
E. default zero one two

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation Although it is normally placed last, the default statement does not have to be the last item of the case block. Because there is no case label found matching the expression, the default label is executed and the code continues to execute until it encounters a break.

**Question 92 (ID: # 17052)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Automatic Data Conversion |

Which of the following statements need to be true in order for the implicit assignment conversion for primitives to take place?

A. The source type and destination type must be the same.
B. The source type and destination type must be compatible.
C. The source type size must be larger than the destination type.
D. The source type must be the same size as the destination type.
E. The destination type size must be larger than the source type.

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation When you assign one type of data to a variable of another type, Java will convert the data to the new variable type automatically, if they are compatible.
For establishing compatibility among primitives, Java uses the following conversion mechanisms:
- identity conversions;
- widening of primitive types; and
- narrowing of primitive types.

**Question 93 (ID: # 17053)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Java Language

**Description** Operator Precedence

Which of the following statements are true for the code below?

```
public class Operators {
	public static void main(String args[])
	{
		String s = "java";
		int x = 4;
		boolean b = true;
		s = s+= x-- - 2 * 3 +
		((Object)s instanceof String ? 3 << 9 / 6 & --x :7 ^ 8 % ++x) | 9;
		System.out.println(s);
	}
}
```

A. The class will compile.

B. The class will throw an exception.

C. The class will print "java11".

D. At the end, the value of x will be 2.

E. The class will print "java5".

Correct Answer A ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation The expression is correct and the program will compile correctly. The issue here is the precedence of the operators. In the expression, x-- has the highest precedence, so it reduces the expression to s=s+=4-2*3 + .. and decreases x to 3. The next operator is *, and after that - and +, which leads to s=s+= -2 + ((Object)s instanceof String?.. . The object s is casted to Object but it remains a String, so the ternary operator will evaluate the left expression 3 << 9 /6 & --x . x is decreased to 2, then used in the remaining expression 3 << 9/6 & 2 , which will execute the (integer) division 3 << 1 & 2 . Then, the shift operator has higher precedence and will reduce the expression to 6 & 2. 6 is binary 110 and 2 is 10, so 110 & 010 will return binary 10 which is decimal 2. The whole expression now reads s=s+= -2 + 2 | 9 , which naturally gets to s=s+=9 , so the program will print "java9" . Therefore, Options A and D are correct, since after the two applications of the -- operator, x is now 2.

**Question 94 (ID: # 17054)**

**Subject** Java 2 Standard Edition 1.5

| | |
|---|---|
| **Subtopic** | Java Language |
| **Description** | Method Signature and Data Promotion |

Which of the following statements are true regarding the code below?

```
public class MethodSignature {

	public static void main(String args[]) {
		add(1/2);
		add(3.14);
	}

	public static void add(float f) {
		System.out.println("f+PI="+ f+3.14 );
	}
}
```

A. It will print "f+PI=3.14" and "f+PI=6.2800001."

B. It will print "f+PI=3.64" and "f+PI=6.2800001."

C. It will print "f+PI=0.03.14" and "f+PI=3.143.14."

D. It will print "f+PI=0.53.14" and "f+PI=3.143.14."

E. The code will not compile.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The division of 1/2 yields an integer value of 0, not 0.5 as you might expect. The evaluation of the expression in the System.out.println call uses the overloaded + to concatenate a number to a string. Option C is false because the code will not compile in the end. The signature of the add method requires a float value. However, if you pass it 3.14 , which is by default double and larger than that float, the compiler doesn't know how to cast it, and it will yield "add(float) in Test cannot be applied to (double)".

**Question 95 (ID: # 17055)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Return Types |

Which of the following lines, when inserted at the commented line in the code below, will not compile?

```
short myFunc (int x, int y) {
	//line to be inserted
}
```

A. return (byte)(y/x);

B. return (short)(x/y);

C. return (char)x;

D. return (byte)3.14D;

E. return (short)y/17;

Correct Answer C ;E ;

User Answer

| | |
|---|---|
| Elapsed Time (seconds) | 0 |
| Explanation | Option C won't compile because char is unsigned and the compiler may complain of a possible loss of precision . Option E will not compile either, because the cast to short applies only to y, not to the whole expression. Therefore, the expression has an int result, and a possible loss of precision. The other possibilities force a cast to short or to byte (which will be silently promoted to short ), so it will raise no objections from the compiler. |

**Question 96 (ID: # 17056)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Array Assignation |

Which of the following lines may be inserted at the specified location in the code below, with the program still compiling?

```
public class Test {
      public static void main(String args[]) {
            short [][] b = new short[3][5];
            short [][] big = new short[4][4];
            short b3 = 8;
            short b2 [][][][] = new short[4][5][4][4];
            //insert code here
      }
}
```

A. b2[1][3] = big;

B. b[1][0] = b3;

C. b2[2][1][1] = b[1][0];

D. b2[1][1][3] = new short[4] {1, 2, 3, 4};

E. b2[2][1] = new byte[4][4];

| | |
|---|---|
| Correct Answer | A ;B ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The multi-dimensional arrays are arrays of arrays, so all the assignments are legal as long as the dimensionality and type are respected. The assignment at Option C does not respect the dimensionality, and the line at Option E does not respect the type (the fact that a byte is silently promoted to a short , doesn't make it possible for a byte array to be promoted to a short array). The problem with Option D is that the compiler does not allow you to specify the size of an initialized array. The correct form would have been new short[] {1, 2, 3, 4}; . Therefore , only Options A and B are correct. |

**Question 97 (ID: # 17057)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Assignment Side Effect |

What will the result be when the code below is run?

```
boolean b = false;

if (b = true) { System.out.println("true");}
      else { System.out.println("false"); }

if (b = false) { System.out.println("false"); }
      else { System.out.println("true"); }
```

A. true false
B. false false
C. false true
D. true true
E. The code won't compile.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation First, you must not confuse the = and == operators, so Option A is not correct. Usually an assignment is not allowed in if condition, but we must remember that an assignment's side effect is the result of the assignment. So, assigning a Boolean (literal or variable) to a Boolean variable results in a Boolean, which is okay as an if condition. So the assignments will evaluate to true in the first case and to false in the second. Thus, Option D is correct.

**Question 98 (ID: # 17058)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Language
**Description** Shift Operators

Which two of the following statements will evaluate to the same value?

A. 131 >> (3 & 6)
B. 256 >>> 5
C. 2^5
D. (8 % -1) << 2
E. (4 >> 1) * (16 << 2) / (1 << 2)

Correct Answer A ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation The expressions A through E evaluate to 32, 8, 7, 0, and 32, respectively. Options A and E are correct, since they both evaluate to 32.

**Question 99 (ID: # 17059)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Language

**Description** Char Declaration

Which of the following are valid declarations of a `char` primitive?

A. char c1 = '\ubeef';
B. char c2 = 'cafe';
C. char c3 = 043312;
D. char c4 = 0xbabe;
E. char c5 = '\xcafe';

Correct Answer A ;C ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation Option A is correct because it's a UNICODE representation of a char . Option C is correct because the octal number fits in 16 bits, so it is the hexa number of Option D. Option B is incorrect because it is not a char literal. Option E is not a UNICODE representation of a char , despite the escape sign.

**Question 100 (ID: # 17060)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Java Language
**Description** Short-Circuit Operators

What will the result be when the code below is run?

```
int i=3;
if (++i == 4 || i++ >6 || ++i > 5) {
      if (++i < 10 | i++ > 0) {
            System.out.println(i);
      }
}
```

A. 6
B. 8
C. 7
D. Nothing will be printed.
E. The code will not compile.

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation A difference should be noticed between the operators | and ||, which basically do the same thing: act as the logical OR. When you have A || B , and A == true , the expression is no longer evaluated, assuming that the whole expression is true. Hence, in the code above, the variable i is no longer incremented in the second and third expression of the outermost if. However, | forces the evaluation of all the expressions. Thus, the code will compile correctly and will print "6."

**Question 101 (ID: # 17061)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Java Language |
| **Description** | Java Keywords |

Which of the following are literals and NOT valid Java keywords?

A. try
B. null
C. false
D. true
E. break

Correct Answer B ;C ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation The following reserved words are literals rather than keywords: null , false , and true .

**Question 102 (ID: # 17062)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Overloading Rules |

Which of the following statements about overloading methods are true?

A. Overloading methods may change the return type.
B. Overloading methods may not declare new or broader checked exceptions.
C. Overloading methods can change the access modifier.
D. A final method cannot be overloaded.
E. An overloading method must change its parameters list.

Correct Answer A ;C ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation Options A, C, and E are correct. An overloading method is a completely new method that is completely unrelated to the overloaded method(s).
An overloading method may use any signature or access modifier. The only thing it must do is change the parameter list, since this may confuse the compiler when a call is made to one of the methods identified by the same name.
The signature of a method includes the name of the method and the number and types of formal parameters to the method. A class cannot declare two methods with the same signature, or a compile-time error will take place . If two methods of a class own the same name but have different signatures (whether they are both declared in the same class, both inherited by a class, or one is declared and one is inherited), then the method name is said to be overloaded. This fact never results in a compile-time error, of itself. There is no necessary relationship between the return types or between the throws clauses of two methods with the same name but different signatures. Methods are overridden on a signature-by-signature basis.

**Question 103 (ID: # 17063)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Overloaded Method Binding |

What will be the result be when the code below is run?

```
public class Test {
      public static void main(String args[]) {
            Test test = new Test();
            Animal a = new Tiger();
            Tiger t = new Tiger();
            test.identify(a);
            test.identify(t);
      }
      public void identify(Animal a) {
            System.out.println("animal");
      }
      public void identify(Tiger t) {
            System.out.println("tiger");
      }
}

class Animal {}
class Tiger extends Animal {}
```

A. animal animal
B. tiger tiger
C. tiger animal
D. animal tiger
E. The code will not compile.

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Option D is correct. The code will compile, because the rules for overloaded methods apply. It will print animal tiger because the decision to link an overloaded method is made at compile time, based on the reference type. |

**Question 104 (ID: # 17064)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Overloaded Constructors |

What will be the result be when the code below is run?

```
public class Test {
	public static void main(String args[]) {
		Dog dog = new Dog();
	}
}

class Dog {
	public Dog() {
		Dog("Max");
	}
	public Dog(String name) {
		System.out.println(name);
	}
	public void Dog(String n) {
		System.out.println("Azorel");
	}
}
```

A. Max
B. Azorel
C. The code will print nothing.
D. The code will fail to compile.
E. A NoSuchMethodException will be thrown at runtime.

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation Option B is correct. Creating a method with the same name as the constructor (class) is perfectly legal. The invocation in the default constructor is to the method and not to the overloaded constructor.

**Question 105 (ID: # 17065)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** OOP Elements
**Description** Default Constructors

The compiler will provide a default constructor for the code below.

```
class Shark {
	void Shark() { }
}
```

Correct Answer TRUE

User Answer

Elapsed Time (seconds) 0

Explanation The compiler will provide a default constructor for the class, since it has no declared constructors. It only contains a method, not a constructor, although it has the same name as the class.

**Question 106 (ID: # 17066)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Benefits of Encapsulation |

Which of the following are not benefits of encapsulation?

| | |
|---|---|
| A. | clarity of code |
| B. | the ability to add functionality later |
| C. | code efficiency |
| D. | access modifiers become optional |
| E. | code maintenance and refactoring is easier |

| | |
|---|---|
| Correct Answer | C ;D ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Options C and D are correct. Encapsulation helps you make the code more readable, extensible, and maintainable, but it does not necessarily make it more efficient. Encapsulation is about object design and does not affect language rules such as the use of modifiers. |

**Question 107 (ID: # 17067)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Method-Local Inner Class Modifiers |

Which of the following are legal modifiers for a method-local inner class?

| | |
|---|---|
| A. | public |
| B. | synchronized |
| C. | private |
| D. | abstract |
| E. | final |

| | |
|---|---|
| Correct Answer | D ;E ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Options D and E are correct. A method-local inner class can be declared abstract or final , but access modifiers like public or protected don't have a meaning in this context, since the class is visible only within the method. Synchronizing a class is also pointless. |

**Question 108 (ID: # 17068)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Static Inner Class Rules |

Which of the following statements are true for static nested classes?

A. A reference to an instance of the enclosing class is needed in order to instantiate.

B. All methods and fields must be declared static .

C. It must extend the enclosing class.

D. It must be instantiated in a static block or method.

E. none of the above

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation Option E is correct. A reference to an instance is required only for regular inner classes (the static classes being nested rather than inner). A static nested class doesn't have any restrictions for member modifiers, and it doesn't need to extend anything in particular.

**Question 109 (ID: # 17069)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** OOP Elements

**Description** Anonymous Classes

Which of the following statements are true for an anonymous inner class?

A. It can extend only one class and implement only one interface.

B. It can implement multiple interfaces if it doesn't extend a class.

C. It can extend only one class or implement only one interface.

D. It has access to method local variables.

E. It may be declared static when it is inside a method.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The syntax allows for an anonymous inner class to extend one class or implement one interface, but not both. When it is instantiated inside a method, it has access only to final local variables and cannot be declared static .

**Question 110 (ID: # 17070)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** OOP Elements

**Description** Equals Method Overriding

What will the result be when the code below is run?

```
1. public class Test {
2.       public static void main(String args[]) {
3.             Object obj = new Object() {
4.                   public boolean equals(Object o) {
5.                         return true;
6.                   }
7.             }
8.             System.out.println(obj.equals("Hawkeye"));
9.       }
10. }
```

A. Hawkeye

B. true

C. It will fail to compile at Line 3.

D. It will fail to compile at Line 4.

E. It will fail to compile at Line 7.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The code would have been legal and would have printed true , if not for the missing semicolon at Line 7.

**Question 111 (ID: # 17071)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** OOP Elements

**Description** Instantiating Anonymous

Which of the following statements is true about the code below?

```
1. public class Test {
2.       public static void main(String args[]) {
3.             GenericNumber number = new GenericNumber() {
4.                   public int toBase10() {
5.                   //converting number to base 10 code..
6.                   }
7.             };
8.             number.setDigits("345523");
9.             System.out.println(number.toBase10());
10.      }
11. }
12.
13. abstract class GenericNumber {
14.      private String digits;
15.      private int base;
16.      public GenericNumber(int base) {
17.            this.base = base;
18.            System.out.println("generic number");
19.      }
20.      public void setDigits(String digits) {
21.            this.digits = digits;
22.      }
23.      public abstract int toBase10();
24. }
```

A. It will fail to compile at Line 3.

B. It will fail to compile at Lines 3 and 4.

C. It will fail to compile at Lines 3 and 13.

D. It will fail to compile at Line 23.

E. It will compile and print generic number , and the converted number to base 10.

Correct Answer A

User Answer

Elapsed Time (seconds) 0

Explanation The anonymous class cannot find an argument-less constructor in the super class to call. It will fail with errors at Line 3. Instantiating an anonymous class extending an abstract class is perfectly legal.

**Question 112 (ID: # 17072)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** OOP Elements

**Description** Method Signatures

Which of the following method declarations would be valid in an interface declaration?

A. public static void main(String[] arguments);

B. protected void main(String[] wow);

C. boolean setFlags(Boolean [] test []);

D. private void dealCards(Hands[] hands);

E. public final float getPoints();

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation Option C is correct. Methods in an interface declaration can only be public and abstract . The protected access modifier cannot be used for fields and methods within an interface. So methods in an interface declaration cannot be static (Option A), protected (Option B), private (Option D), or final (Option E).

**Question 113 (ID: # 17073)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** OOP Elements

**Description** Method Overriding and Overloading

For the code below, which of the following methods, if inserted in the `Point3D` declaration at the specified location, will cause the code to compile?

```
public class Point2D {
      int move(float angle, int distance) {
            // ...
      }
}

public class Point3D extends Point2D{
      //insert code here
}
```

A. private int move(float angle, int dist) {return 1;}

B. protected int move(float angle, int dist) {return 2;}

C. private double move(int angle, float dist) {return 3;}

D. Integer move(float angle, int dist) {return 4;}

E. long move(float angle, int d) {return 5;}

Correct Answer B ;C ;

User Answer

Elapsed Time (seconds) 0

Explanation Option B is correct because the protected modifier is less restrictive than the default modifier. This is a legal override, as is Option C, an overloading of the method. Option A represents an illegal overriding because of the restrictive private modifier. Options D and E are illegal overloadings, because they conflict with the overloaded method in the arguments list.

**Question 114 (ID: # 17074)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** OOP Elements

**Description** Constructor Rules

Which of the following statements regarding constructors are true?

A. A constructor may throw an exception.

B. A constructor must return void .

C. A default constructor is provided by the compiler if no constructors have been declared.

D. The compiler automatically inserts the super(); call if the constructor does not explicitly call super() .

E. The constructors with arguments are inherited by the subclasses.

Correct Answer A ;C ;

User Answer

Elapsed Time (seconds) 0

Explanation Options A and C are correct. A constructor is free to throw an exception. If no constructors have been explicitly written, one will be supplied by the compiler. A constructor doesn't return any kind of value, and it is not inherited by the subclasses. The compiler automatically inserts a call to super(); , except when a call to super() or this() is made explicitly on the first line of the implementation.

**Question 115 (ID: # 17075)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Method-Local Inner Classes |

A method-local inner class has access to the method's variables, but it doesn't have access to the outer class member variables.

| | |
|---|---|
| Correct Answer | FALSE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | A method-local inner class can access only the final method variables, and it does have access to outer class member variables. Therefore, the statement is false. |

**Question 116 (ID: # 17076)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Interface Modifiers |

Which of the following modifiers can be specified in an interface declaration?

A. protected
B. abstract
C. static
D. final
E. synchronized

| | |
|---|---|
| Correct Answer | A ;B ;C ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | An interface declaration might include the following interface modifiers: public protected, private, abstract, static, and strictfp . Not all modifiers are applicable to all types of interface declarations. The access modifiers protected and private relate only to member interfaces inside a directly enclosing class declaration. The access modifier static relates just to member interfaces. |

**Question 117 (ID: # 17077)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | OOP Benefits |

Which of the following are the benefits of Object Oriented Programming?

A. clarity of code
B. late-binding
C. inheritance
D. reusability
E. encapsulation

| | |
|---|---|
| Correct Answer | C ;D ;E ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Clarity is a benefit of a clean programming style. Late-binding is compiler-specific and not a quality of OOP. Inheritance, code reusability, encapsulation, and polymorphism are benefits of OOP. |

**Question 118 (ID: # 17078)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Inner Class Definition |

A(n) _____ class needs an enveloping class instance in order to be instantiated.

| | |
|---|---|
| A. | static |
| B. | nested |
| C. | private |
| D. | inner |
| E. | anonymous |

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | An inner class needs an instance of the outer class in order to be instantiated. |

**Question 119 (ID: # 17079)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | OOP Elements |
| **Description** | Complete Encapsulation |

For the code below, which of the following statements are true?

```
class Doll {
      private String name;
}
class RussianDoll extends Doll {
      private RussianDoll smallerDoll;
}
```

| | |
|---|---|
| A. | RussianDoll "is a" Doll . |
| B. | RussianDoll "has a" Doll . |
| C. | RussianDoll "has a" RussianDoll . |
| D. | class Doll is not entirely encapsulated. |
| E. | smallerDoll can't have access to the name property. |

| | |
|---|---|
| Correct Answer | A ;B ;C ;D ;E ; |

User Answer

Elapsed Time (seconds) 0

Explanation All of the statements are true. The class Doll is not entirely encapsulated because it doesn't provide any methods for accessing the property name.

**Question 120 (ID: # 17149)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.io

**Description** Readerobject Creation

Which of the following statements represents the method for creating a `InputStreamReader` instance during an `InputStream`?

A. Use the static method createReader(..) of InputStream .

B. Use the static method createReader(..) of Reader .

C. Create an InputStreamReader and pass it the InputStream .

D. Pass the InputStream to the Reader constructor.

E. Pass the InputStream to the Object constructor.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The InputStreamReader class provides a constructor which accepts an InputStream as an argument and creates Reader objects.

**Question 121 (ID: # 17150)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.io

**Description** Writer Attributes

Which of the following statements about the Writer class are true?

A. A Writer class can be used to write characters in an output stream using different encodings.

B. The Writer class has overloaded writing methods for all of the Java primitive types.

C. The Writer class has special methods for writing objects on an output stream.

D. Writer instances can be used to write strings that hold Unicode chars .

E. The Writer class extends the OutputStream .

Correct Answer A ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation The Writer class is oriented toward writing characters in different encodings, including Unicode. It has no mechanism for writing primitives or objects. The Writer is a direct subclass of Object , and doesn't implement the OutputStream .

**Question 122 (ID: # 17151)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | LineNumberReader Usage |

What will be the result be when the code below is run?

```
1. import java.io.*;
2. public class TestIO {
3.       public static void main(String [] args) throws IOException {
4.             StringReader stringin = new StringReader("test");
5.             LineNumberReader in = new LineNumberReader(stringin);
6.             PrintWriter out = new PrintWriter(System.out);
7.             out.println(in.readLine());
8.             out.flush();
9.       }
10. }
```

A. 1: test

B. 1 test

C. test

D. It will fail to compile at Line 5.

E. It will fail to compile at Line 6.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The LineNumberReader doesn't add line number information to the entry content, so it prints only test .

**Question 123 (ID: # 17152)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | Unicode and Char Size |

Which of the following statements are true?

A. All Unicode characters are represented on 16 bits.

B. For representing ASCII characters, only 7 bits are needed.

C. UTF-8 are Unicode characters represented on one byte only.

D. UTF-16 needs two bytes to represent Unicode characters.

E. From a two-byte Unicode character, the less significant byte represents ASCII code.

Correct Answer A ;B ;

User Answer

Elapsed Time (seconds) 0

| | |
|---|---|
| Explanation | UTF-8 and UTF-16 are representation formats with variable length. The less significant byte represents ASCII code only for the Unicode characters from page 0. |

**Question 124 (ID: # 17153)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | File Class Usage |

Which of the following statements are true for the File class?

| | |
|---|---|
| A. | A File object can be used to change the current active folder. |
| B. | A File object can be used to access files from the active folder. |
| C. | When a File object is created, a file or folder is created on the local file system. |
| D. | The File class has methods for changing file or folder attributes, such as visibility, owner, and access rights. |
| E. | When a File instance is garbage collected, the corresponding file or folder is deleted from the local file system. |

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The File object can be used to access files or folders from the active folder, but doesn't allow for file modifications. When an instance is created, there is no need for a corresponding file or folder to be tightly-coupled with the instance. Thus, files and folders are not created and deleted like instances. |

**Question 125 (ID: # 17154)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | Serializable Methods |

Which of the following methods are declared by the Serializable interface?

| | |
|---|---|
| A. | public void write(OutputStream os); |
| B. | public void save(); |
| C. | protected void save(); |
| D. | public void writeObject(Object o); |
| E. | none of above |

| | |
|---|---|
| Correct Answer | E |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The Serializable interface doesn't declare any methods. It is used only as a marker for the objects eligible to be serialized. |

**Question 126 (ID: # 17155)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | File mkdirs() |

The method `mkdirs()` is used to _____.

| | |
|---|---|
| A. | create a directory |
| B. | create a directory including any necessary but nonexistent parent directories |
| C. | set the last-modified time of the directory |
| D. | create a directory with the name "New Folder" |
| E. | set the user account that last modified the directory |

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | In Java, the File class provides a method called mkdirs() to create directories on all the platforms. |

**Question 127 (ID: # 17156)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | Methods Available for an Instance of DataInputStream |

Which of the following are valid ways to create `DataInputStream` streams?

| | |
|---|---|
| A. | new DataInputStream(); |
| B. | new DataInputStream("in.dat","r"); |
| C. | new DataInputStream("in.dat"); |
| D. | new DataInputStream(new File("in.dat")); |
| E. | new DataInputStream(new FileInputStream("in.dat")); |

| | |
|---|---|
| Correct Answer | E |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | To create the instance of a DataInputStream class, you have to pass FileInputStream or DataInput(System.in) . Hence, the constructor DataInputStream(new FileInputStream("in.dat")) is a valid one. |

**Question 128 (ID: # 17157)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | FilterInputStream Subclasses |

Which of the following classes are subclasses of `FilterInputStream`?

| | |
|---|---|
| A. | CheckedInputStream |

B. LineNumberInputStream
C. DigestInputStream
D. BufferedInputStream
E. CipherInputStream

Correct Answer A ;B ;C ;D ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation FilterInputStream contains other input streams, which it uses as its basic source of data. The data may be transformed along the way, or additional functionality may be added. The class FilterInputStream itself simply overrides all methods of InputStream with versions that pass all requests to the input stream contained. Subclasses of FilterInputStream may further override some of these methods and may also provide additional methods and fields. Direct subclasses of FilterInputStream are the following: BufferedInputStream, CheckedInputStream, CipherInputStream, DataInputStream, DigestInputStream, InflaterInputStream, LineNumberInputStream, ProgressMonitorInputStream, and PushbackInputStream .

**Question 129 (ID: # 17158)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.io
**Description** RandomAccessFile

What will the result be when the code below is run?

```
import java.io.*;

public class RATest {
      public static void main(String[] a) throws IOException {
            RandomAccessFile file = new RandomAccessFile("test.dat","rw");
            file.writeBoolean(true);
            file.writeInt(123456);
            file.writeInt(7890);
            file.writeLong(1000000);
            file.writeInt(777);
            file.writeFloat(.0001f);
            file.seek(5);
            System.out.println(file.readInt());
            file.close();
      }
}
```

A. 123456
B. 7890
C. 1000000
D. 777
E. It will fail to compile.

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation The Boolean value fills the first byte, and the next four bytes are filled with the int value 123456 . Therefore, the next int , 7940 , is read and displayed.

**Question 130 (ID: # 17159)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | StreamTokenizer Role |

The _____ class takes an input stream and parses it into fragments, allowing the fragments to be read one at a time. The parsing process is controlled by a table and a number of flags that can be set to various states.

| | |
|---|---|
| A. | StreamParser |
| B. | BlockInputStream |
| C. | StreamTokenizer |
| D. | BlockReader |
| E. | StringParser |

| | |
|---|---|
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The StreamTokenizer class takes an input stream and parses it into tokens, allowing the tokens to be read one at a time. The parsing process is controlled by a table and a number of flags that can be set to various states. The stream tokenizer can recognize identifiers, numbers, quoted strings, and various comment styles.<br>Each byte read from the input stream is regarded as a character in the range '\u0000' through '\u00FF'.<br>The character value is used to look up five possible attributes of the character: white space, alphabetic, numeric, string quote, and comment character. Each character can have zero or more of these attributes. |

**Question 131 (ID: # 152279)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |
| **Description** | Assertion Statement |

In the assertion statement below, `Expression2` is a Boolean expression.

```
assert Expression1 : Expression2 ;
```

| | |
|---|---|
| Correct Answer | FALSE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Expression1 represents a Boolean expression.<br>Expression2 represents an expression that has a value. (It mustn't be an invocation of a method that is declared void .) |

**Question 132 (ID: # 152282)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.io |

| **Description** | Available Methods of the InputStream Class |
|---|---|

The code below is guaranteed to read all of the remaining bytes from the given input stream.

```
int n = in.available();
byte buf = new byte[n];
in.read(buf);
```

| Correct Answer | FALSE |
|---|---|
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The available method of the InputStream class and its subclasses does not automatically return the maximum number of bytes that might be read with no blocking. The code above is not guaranteed to read all of the remaining bytes from the known input stream. Likewise, the ready method of the Reader class and its subclasses might return false , even if the stream is prepared to be read. |

**Question 133 (ID: # 17116)**

| **Subject** | Java 2 Standard Edition 1.5 |
|---|---|
| **Subtopic** | Package java.lang |
| **Description** | Wrappers |

Which of the following is not a wrapper class?

| A. | Boolean |
|---|---|
| B. | Character |
| C. | Short |
| D. | String |
| E. | Double |

| Correct Answer | D |
|---|---|
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | String doesn't fit because the others are primitive wrappers and String is not, although it is immutable like they are. |

**Question 134 (ID: # 17117)**

| **Subject** | Java 2 Standard Edition 1.5 |
|---|---|
| **Subtopic** | Package java.lang |
| **Description** | String Concatenation |

What will the result be when the code below is run?

```
public class Question {
        public static void main(String[] args) {
                String s1 = "abc";
                String s2 = "def";
                String s3 = s1.concat(s2.toUpperCase());
                System.out.println(s1+s2+s3);
        }
}
```

A. abcdefabcdef
B. abcdefabcDEF
C. abcdefABCDEF
D. abcDEFabcDEF
E. The program will not compile.

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation The program will compile and print abcdefabcDEF , because the String objects are immutable.

**Question 135 (ID: # 17118)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.lang
**Description** String Methods

Which of the following are String methods?

A. delete()
B. intern()
C. append()
D. reverse()
E. replace()

Correct Answer B ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation Of the options provided, only replace() and intern() are String methods.

**Question 136 (ID: # 17119)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.lang
**Description** Subclasses of Number

Which of the following classes extend `java.lang.Number`?

A. Double
B. Short
C. BigInt
D. Int
E. Byte

Correct Answer A ;B ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation The classes Int and BigInt don't exist. The rest all extend java.lang.Number .

**Question 137 (ID: # 17120)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.lang

**Description** Math Functions

What will the result be when the code below is run?

```
1. public class Test {
2.       public static void main(String[] args) {
3.             double v1 = 123.45;
4.             long v2 = 12345;
5.             System.out.println(Math.max(v1,v2));
6.             System.out.println(v1 = Math.floor(v1));
7.             System.out.println(Math.ceil(v2));
8.             System.out.println(v2 = Math.rint(v1));
9.       }
10.}
```

A. 12345.0 123.0 12345.0 123.0
B. It will fail to compile at Line 5.
C. It will fail to compile at Line 6.
D. It will fail to compile at Line 7.
E. It will fail to compile at Line 8.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The long value of v2 is promoted to double in Lines 5 and 7, which is fine for the compiler.
The side-effect of the assignment is accepted by println(..) as a double value, but in Line 8, the compiler will notice a possible loss of precision and it will stop compiling.

**Question 138 (ID: # 17121)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.lang

| | |
|---|---|
| **Description** | Square Root of a Negative Number |

What will the output be when the code below is run?

```
System.out.println(Math.sqrt(-17));
```

A. It will not compile.
B. It will throw an ArithmeticException .
C. It will print "-Infinity" (Double.NEGATIVE_INFINITY).
D. It will print "NaN".
E. It will throw an IllegalArgumentException .

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Extracting the square root of a negative number will result in a NotANumber result, and no exception will be thrown. |

| | |
|---|---|
| **Question 139 (ID: # 17122)** | |
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.lang |
| **Description** | Math Suite |

What will the output be when the code below is run?

```
1. public class Test {
2.       public static void main(String[] args) {
3.             double pi = Double.POSITIVE_INFINITY;
4.             double notNum = Double.NaN;
5.             if(notNum == notNum) {
6.                   System.out.println("equal");
7.                   } else {
8.                         System.out.println("unequal");
9.                         }
10.            System.out.println(16.43d/-0.0);
11.            System.out.println(16/-0);
12.      }
13. }
```

A. It will print "unequal -Infinity," then throw an exception.
B. It will print "equal -Infinity -Infinity."
C. It will print "equal -Infinity," then throw an exception.
D. It will not compile due to Line 10 or Line 11.
E. It will print something else.

| | |
|---|---|
| Correct Answer | A |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation Two NaN numbers are not equal. Dividing by 0 only works for floating-point numbers. Therefore, it will first compute at Line 10, but Line 11 will throw an ArithmeticException . The result of dividing by -0.0 is -Infinity .

**Question 140 (ID: # 17123)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.lang
**Description** Math Class Declaration

What is the declaration of the Math class from the `java.lang` package, since JDK1.3+?

A. public final class Math
B. public static final class Math
C. public final strictfp class Math
D. public abstract class Math
E. public final class Math extends StrictMath

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The Math class is final, and it tells the compiler that it uses strict floating point representation. StrictMath is a delegate class internally used by Math , but the latter doesn't extend the former.

**Question 141 (ID: # 17124)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.lang
**Description** Math.random()

Which of the following statements is true about the static method `Math.random()`?

A. It only returns values that are greater than 0.
B. It takes one double argument == the maximum to return -1.
C. It may throw an IllegalArgumentException .
D. It returns values greater than or equal to 0.
E. It is declared final so that it can't be overridden.

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation The Math.random method doesn't take any arguments, therefore it can't throw IllegalArgumentException . It can't be overridden since it is in the Math class, so it doesn't need to be declared final . The result of the function is a random number in range (0, 1) .

**Question 142 (ID: # 17125)**

**Subject** Java 2 Standard Edition 1.5

| | |
|---|---|
| **Subtopic** | Package java.lang |
| **Description** | String and Wrappers |

Which of the following statements are true about wrapper classes or String class?

A. If x and y refer to different wrapper classes, then x == y can sometimes be true.

B. If x and y refer to different wrapper classes, then the fragment x.equals(y) will cause a compilation error.

C. If x and y are String references, and if x.equals(y) , then x == y is true.

D. If x, y, and z refer to instances of the wrapper classes and x.equals(y) is true and y.equals(z) is true then x.equals(z) will also be true.

E. If x and y are String references, and if x == y, then x.equals(y) is true.

Correct Answer D ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation Option D is true based on the contract that equals() is transitive. Option E is true because x and y are referring to the same object. and equal is by contract consistently reflexive. All of the other statements are false.

**Question 143 (ID: # 17126)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.lang |
| **Description** | currentTimeMillis |

The method `currentTimeMillis ()` is a method of the _____ class in the `java.lang` package.

A. Thread

B. Date

C. Object

D. System

E. ThreadGroup

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation The method currentTimeMillis is a static method in the System class. It returns the current date and time, represented as the number of milliseconds passed since January 1st, 1970.

**Question 144 (ID: # 17127)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.lang |
| **Description** | ThreadGroup in the Java.lang Package |

Which of the following are `ThreadGroup` constructors?

A. ThreadGroup (String groupname)

B. ThreadGroup (String groupname , int numbers_of_threads)

C. ThreadGroup (int number_of_threads)

D. ThreadGroup (String groupname, Object obj)

E. ThreadGroup (ThreadGroup parentobject, String groupname)

Correct Answer A ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation ThreadGroup defines two constructors: ThreadGroup (String groupname) and ThreadGroup(Threadgroup parentob,String groupname) . For both forms, the groupname specifies the name of the thread group. The first version creates a new group that has the current thread as its parent. In the second form, the parent is specified by the parent object.

**Question 145 (ID: # 17128)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.lang

**Description** Default Packages

Which of the following is the default package imported by any Java class?

A. java.util

B. java.lang

C. java.lang.reflect

D. java.io

E. java.math

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation The package java.lang is automatically imported into all programs. It contains classes and interfaces that are fundamental to virtually all of Java programming. It is Java's most widely used package.

**Question 146 (ID: # 17129)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.lang

**Description** Java.lang Subpackages

Which of the following are standard subpackages of `java.lang`?

A. java.lang.utils

B. java.lang.ref

C. java.lang.reflect

D. java.lang.math

E. java.lang.gc

Correct Answer A ;B ;

User Answer

Elapsed Time (seconds) 0

Explanation The two subpackages of java.lang are java.lang.ref and java.lang.reflect . The classes in the java.lang.ref package, which was added by version 2.0 of Java, provided more flexible control over the garbage collection process. Reflection is the ability of a program to analyze itself. The java.lang.reflect package provides the ability to obtain information about the fields, constructors, methods, and modification of a class.

**Question 147 (ID: # 17130)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.lang

**Description** Custom java.lang Subpackage

What will happen if valid code is compiled and executed as a class in a `java.lang.my` custom subpackage?

A. The compiler will fail to compile in the standard subpackage.

B. The compiler will compile successfully, but a warning will appear about possible package conflicts.

C. The code will compile and then throw a SecurityException .

D. The code will compile and then throw an IllegalAccessException .

E. The code will compile and run, with no problems.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The code will compile with no warnings, but the ClassLoader will throw a SecurityException : Exception in thread "main" java.lang.SecurityException : Prohibited package name: java.lang.my .
This will prevent any attempts to replace system classes with custom classes that will corrupt the system.

**Question 148 (ID: # 17131)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Package java.lang

**Description** Cloneable

A class implements the _____ interface to indicate to an object's method that it is legal for that method to make a field-for-field copy of instances of that class.

A. Serializable

B. Externalizable

C. Copyable

D. Reproduceable

E. Cloneable

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation A class implements the Cloneable interface to indicate to the Object.clone() method that it is legal for that method to make a field-for-field copy of instances of that class. Invoking an object's Clone method on an instance that does not implement the Cloneable interface results in the CloneNotSupportedException being thrown.

**Question 149 (ID: # 152227)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.nio
**Description** Byte Buffer Order

The order of a newly-created byte buffer is always _____.

A. Force
B. BIG_ENDIAN
C. LITTLE_ENDIAN
D. ShortBuffer
E. ByteOrder

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation Option B is correct. The byte order is employed when reading or writing multibyte values, and when generating buffers that are views of this byte buffer.
public static final ByteOrder BIG_ENDIAN is a constant indicating a big-endian byte order. In this order, the bytes of a multibyte value are ordered from most important to least important.

**Question 150 (ID: # 152228)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.nio
**Description** Equal Buffers

Which of the following requirements need to be met in order for two byte buffers to be equal?

A. They must have the same limit.
B. They must have the same element type.
C. They must have the same number of remaining elements.
D. The two series of remaining elements, regarded independently of their starting positions, must be point wise equal.
E. They must have the same capacity.

Correct Answer B ;C ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation Options B, C, and D are correct. public boolean equals(Object ob) - shows whether or not this buffer is equal to a different object. A byte buffer is not equal to any other type of object.

**Question 151 (ID: # 152229)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.nio |
| **Description** | Rewinding |

Which of the following methods leaves the limit unchanged and sets the position for a given buffer instance to zero?

| | |
|---|---|
| A. | limit() |
| B. | flip() |
| C. | position() |
| D. | clear() |
| E. | rewind() |

| | |
|---|---|
| Correct Answer | E |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Option E is correct. rewind() prepares a buffer for re-reading the data that it already includes. It lets the limit be unmodified, and sets the position to zero. |

**Question 152 (ID: # 152232)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.nio |
| **Description** | Buffer |

Which of the following statements about the `java.nio.Buffer` class are correct?

| | |
|---|---|
| A. | A buffer's position is never negative and can be greater than its limit. |
| B. | A buffer's capacity is the number of elements it contains. |
| C. | A buffer's limit is never negative and is never greater than its capacity. |
| D. | A buffer's position is the index of the first element that should not be read or written. |
| E. | The capacity of a buffer is never negative, and it never changes. |

| | |
|---|---|
| Correct Answer | B ;C ;E ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Options B, C, and E are correct. Excluding its content, the fundamental properties of a buffer are its capacity, limit, and position.<br>A buffer's capacity represents the number of elements it includes. The capacity of a buffer is never negative and it never changes.<br>A buffer's limit represents the index of the first element that must not be read or written. A buffer's limit is never negative and is never bigger than its capacity.<br>A buffer's position represents the index of the next element to be read or written. A buffer's position is never negative, and it is never bigger than its limit. |

**Question 153 (ID: # 17132)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | Unsynchronized Indexed Collections |

Which collection class allows you to grow or shrink its size, and provides indexed access to its elements, but its methods are not synchronized?

A. java.util.HashSet
B. java.util.LinkedHashSet
C. java.util.List
D. java.util.ArrayList
E. java.util.Vector

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation All of the collection classes allow you to grow or shrink the size of your collection.
ArrayList provides an index to its elements. The newer collection classes tend not to have synchronized methods. Vector is an older implementation of the ArrayList functionality, and it has synchronized methods. It is slower than ArrayList .

**Question 154 (ID: # 17133)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | TreeSet Iterated |

What will the result be when the code below is run?

```
TreeSet map = new TreeSet();
map.add("one");
map.add("two");
map.add("three");
map.add("four");
map.add("one");
Iterator it = map.iterator();
while (it.hasNext() ) {
      System.out.print( it.next() + " " );
}
```

A. one two three four
B. four three two one
C. four one three two
D. one two three four one
E. The print order is not guaranteed.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation TreeSet ensures that there are no duplicate entries. Also, when it is accessed, it will return elements in a natural order, which for Strings means alphabetical order.

**Question 155 (ID: # 17134)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** Key-Value FIFOs

Which collection class allows you to associate its elements with key values, and allows you to iterate over them in the order of insertion?

A. java.util.ArrayList
B. java.util.LinkedHashMap
C. java.util.HashMap
D. java.util.LinkedHashSet
E. java.util.TreeSet

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation LinkedHashMap is the collection class used for caching purposes. To retrieve LinkedHashMap elements in cached order, use the values() method and iterate over the resulting collection.

**Question 156 (ID: # 17135)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** Collection Subinterfaces

Which of the following interfaces extend the `java.util.Collection`?

A. Dictionary
B. Enumeration
C. SortedSet
D. List
E. BeanContext

Correct Answer C ;D ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation The java.util.Collection is extended by BeanContext, BeanContextServices, List, Set, and SortedSet . Dictionary is an obsolete class that was replaced by Map implementations. Enumeration is not a collection, although it can be used to enumerate elements of a collection.

**Question 157 (ID: # 17136)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | Internationalizable Classes |

Which classes from `java.util` support internationalization?

A. Locale
B. Language
C. ResourceBundle
D. Country
E. Internationalization is not supported implicitly by any standard class.

Correct Answer A ;C ;

User Answer

Elapsed Time (seconds) 0

Explanation The Locale and ResourceBundle classes implicitly support internationalization. The Language and Country classes do not belong to java.util .

**Question 158 (ID: # 17137)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | Observer and Observable |

Which of the following statements about `Observer` and/or `Observable` are true in regards to `java.util`?

A. Observer is an interface.
B. Observable is an interface.
C. When an Observable is updated, it calls its observers' update() method.
D. Observable objects must be GUI components.
E. The hasChanged() method belongs to Observer .

Correct Answer A ;C ;

User Answer

Elapsed Time (seconds) 0

Explanation Observer is an interface and Observable is a class, despite its name, which can be misleading if you assume that it follows the Sun conventions for interface naming. When an Observable is updated, it calls its observers' update() method. The method hasChanged() belongs to the Observable class.

**Question 159 (ID: # 17138)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | TreeMap Iterated |

What will the result be when the code below is run?

```
1. import java.util.*;
2.
3. public class Test {
4.       public static void main(String[] args) {
5.             TreeMap map = new TreeMap();
6.             map.put("un","1");
7.             map.put("dos","2");
8.             map.put("tres","3");
9.             displayMap(map);
10.      }
11.      static void displayMap(Map m) {
12.            Collection c = m.entrySet();
13.            for(Iterator it = c.iterator();it.hasNext();) {
14.                  System.out.println(it.next());
15.            }
16.      }
17. }
```

A. 123
B. un1dos2tres3
C. dos=2tres=3un=1
D. It will fail to compile at Line 13.
E. It will fail to compile at Line 14.

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The elements of a TreeMap are ordered by the value of their keys.

**Question 160 (ID: # 17139)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** Gregorian Calendar

If you have a Gregorian Calendar set to August 31, 2003, and you call the code below, what date will you obtain?

```
roll(Calendar.MONTH,6)
```

A. February 31, 2004
B. February 29, 2004
C. February 28, 2003
D. March 2, 2004
E. June 31, 2003

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation The calendar will be rolled forward six months, which is February. However, February doesn't have 31 days, so the calendar sets the day to the closest possible, which is 28. The YEAR field maintains the value of 2003 , because it is larger than the MONTH field.

**Question 161 (ID: # 17140)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** Properties Superclass

Which of the following is the superclass/superinterface of `java.util.Properties`?

A. Collection
B. List
C. Object
D. Hashtable
E. ResourceBundle

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation Properties extends the Hashtable class, which implements Map, Cloneable and Serializable .

**Question 162 (ID: # 17141)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** TreeSet Advantages

What added advantages does the `TreeSet` class have over the `HashSet` class?

A. The elements of this set are stored internally in the form of a tree.
B. It guarantees constant access times.
C. TreeSet is more thread-safe.
D. It facilitates quick access and stores the elements in a sorted order.
E. It defines constants that are used as attribute keys in the AttributedCharacterIterator returned from Format.formatToCharacterIterator , and as field identifiers in FieldPosition .

Correct Answer A ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation The TreeSet class uses a tree construct for internal storage to implement a set, which means that access times are faster. One big difference between the two is that in a TreeSet class, objects are stored in sorted order. Furthermore, the sorting order for TreeSet objects can also be specified.

**Question 163 (ID: # 17142)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | LinkedList Role |

How does the `LinkedList` interface work?

A. It links the AbstractList into a sequential list.
B. It links the AbstractList into a random list.
C. It links elements in a list, so that every element knows where the other elements are.
D. It links elements in a list, so that every element knows what the other elements are.
E. It links elements in a list, so that each element knows where the next element is.

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The List interface extends the collection function to implement lists of objects. The LinkedList class extends the AbstractSequentialList into a LinkedList in which each element knows where the previous and the next elements are.

**Question 164 (ID: # 17143)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | ListIterator |

The `ListIterator` interface helps to create _____.

A. a list sorted on the basis of arguments passed by the user
B. a multi-directional linked list
C. a collection in which you can move through or cycle through
D. a bidirectional linked list
E. a garbage linked list

Correct Answer D

User Answer

Elapsed Time (seconds) 0

Explanation In Java collections, iterators can only work forward through a list using the next method, thus enabling iteration (or cycling through) the elements of the collection. List iterators, on the other hand, also support the previous method, so they can work backward through a list. This is particularly useful for linked lists, in which each element often must know about the next and previous elements. This is especially useful when you are implementing a buffer that can grow or shrink.

**Question 165 (ID: # 17144)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | StringTokenizer Accessors |

Which of the following are access methods provided by the `StringTokenizer`?

A. getCount()
B. countTokens()
C. hasNext()
D. hasMoreElements()
E. notifyAll()

Correct Answer B ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation The access methods provided by the StringTokenizer include the Enumeration methods hasMoreElements() and nextElement(), hasMoreTokens() and nextToken() , and countTokens() . The countTokens() method returns the number of tokens in the string being passed. The hasMoreElements() method returns true if one or more tokens remain in the string, and returns false if there are none.

**Question 166 (ID: # 17145)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** BitSet

The `BitSet` class is used to create objects that maintain a set of bits.

Correct Answer TRUE

User Answer

Elapsed Time (seconds) 0

Explanation A BitSet class creates a special type of array that holds bit values. This array can increase in size, as needed. The BitSet implements the Cloneable interface. The BitSet class constructors are BitSet() and BitSet(int size) .

**Question 167 (ID: # 17146)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** Dictionary

The Dictionary class has methods that can be used to put and retrieve objects in the dictionary.

Correct Answer TRUE

User Answer

Elapsed Time (seconds) 0

Explanation The get() method is used to retrieve an object from the dictionary based on its key, and the put() method is used to store an object.

**Question 168 (ID: # 17147)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | Comparator Behavior |

When the `Comparator` class is used, the `Compare` method compares two objects and returns _____.

A. a negative integer if object1 is greater than object2
B. true if object1 equals object2
C. object1.hashCode() - object2.hashCode()
D. whichever one is greater
E. a positive integer if object1 is greater than object2

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation The Comparator interface can determine the sorting order used in a TreeSet object. The Compare method compares two objects, object1 and object2 , and returns a negative integer if object1 is less than object2 , 0 if they are equal, and a positive integer if object1 is greater object2 . Overriding this method allows for a custom sorting order.

**Question 169 (ID: # 17148)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | ResourceBundle |

When your program needs a locale-specific resource, such as a String, your program can load it from the _____ appropriate for the current user's internationalization settings.

A. Locale
B. Properties
C. Bundle
D. PropertyBundle
E. ResourceBundle

Correct Answer E

User Answer

Elapsed Time (seconds) 0

Explanation J2SE applications store locale-specific resources, for instance GUI item labels, menu items, and help text in classes known as resource bundles.

**Question 170 (ID: # 152273)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | Level Class |

The Level class defines three standard log levels.

| | |
|---|---|
| Correct Answer | FALSE |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The Level class defines seven standard log levels, which vary from FINEST to SEVERE. |

**Question 171 (ID: # 152274)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | CharSequence Interface |

Which of the following classes implement the `CharSequence` interface?

A. Matcher
B. String
C. Pattern
D. StringBuffer
E. CharBuffer

| | |
|---|---|
| Correct Answer | B ;D ;E ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Options B, D, and E are correct. The CharSequence interface offers uniform, read-only access to many different types of character sequences. You can have the data looked up using different sources. String , StringBuffer , and CharBuffer implement CharSequence , so they are simple sources of data. You can also write your own input source by implementing the CharSequence interface. |

**Question 172 (ID: # 152277)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Package java.util |
| **Description** | Matcher Methods |

Once generated, a matcher may be employed to execute the match operations corresponding to which of the following methods?

A. matches
B. compile
C. lookingAt
D. find
E. flags

| | |
|---|---|
| Correct Answer | A ;C ;D ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |

Explanation Options A, C, and D are correct.
The matches method tries to match the whole input sequence against the pattern.
The lookingAt method tries to match the input sequence, starting at the beginning, against the pattern.
The find method scans the input sequence, searching for a match to the pattern.

**Question 173 (ID: # 152278)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Package java.util
**Description** Matches Method

Which of the following options is equivalent to the statement below?

```
Pattern p = Pattern.compile("a*b");
Matcher m = p.matcher("aaaaab");
boolean b = m.matches();
```

A. Pattern p = Pattern.matches("a*b", "aaaaab");
B. Matcher m = Pattern.compile("a*b", "aaaaab");
C. boolean b = Pattern.matches("a*b", "aaaaab");
D. Pattern p = m.matches("a*b", "aaaaab");
E. Matcher m = p.matcher("a*b", "aaaaab");

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation Option C is correct. A matches method is defined by this class as a convenience for when a regular expression is employed only one time. This method compiles an expression and matches an input sequence against it in a sole invocation. The statement boolean b = Pattern.matches("a*b", "aaaaab"); is equal to the statement in the question, although it is less efficient for repeated matches, since it does not allow the compiled pattern to be reutilized.

**Question 174 (ID: # 152224)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Swing
**Description** selectionForKey

`public int selectionForKey(char aKey, ComboBoxModel aModel)` returns the row that should become selected, given `aKey` and the model. What value is returned if no match is found?

A. -2
B. -1
C. 0
D. 1
E. 2

Correct Answer B

User Answer

Elapsed Time 0

(seconds)

Explanation This returns an int equal to the chosen row, where 0 is the first item and -1 is none .

**Question 175 (ID: # 17104)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Threads

**Description** Advantages of Implementing Runnable

In Java, there are two ways of obtaining multithreading: implementing the `Runnable` interface, or subclassing the `Thread` class . Which of the following are benefits of implementing `Runnable` instead of subclassing `Thread`?

A. The same job can be run more easily by many threads.

B. No thread instance needs to be started.

C. It allows for a more specialized thread-specific behavior.

D. It prevents the developer from overloading the run() method, instead of overriding it.

E. It allows for a faster development, by benefiting from anonymous instantiation.

Correct Answer A ;D ;

User Answer

Elapsed Time (seconds) 0

Explanation Implementing Runnable allows the developer to keep one reference of the job needed to be executed, and that reference may be passed to many Thread instances. Implementing Runnable allows the job instance to implement other interfaces and extend a class that's more appropriate to the business logic of the intended job. Also, the compiler will prevent the developer from overloading the run() method instead of overriding the one in the Thread class, by mistake. However, for each thread that must run, a new instance of Thread is needed, and it must also be explicitly started. As an application design, extending Thread is more appropriate when a more specialized thread-specific behavior is needed. While implementing Runnable , the difference is that the executed task not the thread behavior.
Anonymous is faster to write, but not only interfaces benefit from it. You can extend a class in an anonymous instance as well, so this does not represent a benefit of an interface versus a class.

**Question 176 (ID: # 17105)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Threads

**Description** Thread States

Which of the following are possible states of a thread?

A. alive

B. new

C. blocked

D. stalled

E. deleted

Correct Answer B ;C ;

User Answer

| | |
|---|---|
| Elapsed Time (seconds) | 0 |
| Explanation | A thread can be in one of the following states.<br>- NEW: A thread that has not yet started is in this state.<br>- RUNNABLE: A thread executing in the Java virtual machine is in this state.<br>- BLOCKED: A thread that is blocked waiting for a monitor lock is in this state.<br>- WAITING: A thread that is waiting indefinitely for another thread to perform a particular action is in this state .<br>- TIMED_WAITING: A thread that is waiting for another thread to perform an action for up to a specified waiting time is in this state.<br>- TERMINATED: A thread that has exited is in this state. |

**Question 177 (ID: # 17106)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Threads |
| **Description** | notify() signature |

Which of the following shows the correct signature for the `notify()` method in the Object class?

| | |
|---|---|
| A. | public static void notify() |
| B. | public final native void notify() |
| C. | public static final int notify() |
| D. | public final synchronized int notify() |
| E. | protected static final int notify() |

| | |
|---|---|
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The notify() method is declared as public final native void notify() in interthread communication. The notify() method wakes up the first thread that called wait() on the same object. |

**Question 178 (ID: # 17107)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Threads |
| **Description** | Minimum Priority |

Which of the following represent ways of defining thread properties?

| | |
|---|---|
| A. | HIGH_PRIORITY |
| B. | MAX_PRIORITY |
| C. | DEFAULT_PRIORITY |
| D. | MIN_PRIORITY |
| E. | NORM_PRIORITY |

| | |
|---|---|
| Correct Answer | B ;D ;E ; |
| User Answer | |
| Elapsed Time | 0 |

(seconds)

Explanation There are three ways of defining thread properties.
(i) MAX_PRIORITY
(ii) MIN_PRIORITY
(iii) NORM_PRIORITY

The maximum priorities that can be set for these properties are 10, 1, and 5 respectively.

**Question 179 (ID: # 17108)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Threads
**Description** Wait() Definition

The `wait()` method is defined in the Object class, this means that it is also available in _____.

A. Runnable interface
B. Thread class
C. ThreadGroup class
D. Date class
E. Format class

Correct Answer B ;C ;D ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation Java includes an elegant inter-process communication mechanism, using wait(), notify() , and notifyAll() . These methods are implemented as final methods in the Object class, so all classes have them.

**Question 180 (ID: # 17109)**

**Subject** Java 2 Standard Edition 1.5
**Subtopic** Threads
**Description** Sleep() Arguments

In the `sleep()` method of the `Thread` class, what argument should be passed?

A. minutes
B. seconds
C. milliseconds
D. microseconds
E. CPU clock ticks

Correct Answer C

User Answer

Elapsed Time (seconds) 0

Explanation There are two arguments that can be passed to a sleep() method. They include (long ml) or milliseconds in terms of a long data type used for declaring long integer variables and returned values. The other is (long ml, int ns) which is milliseconds in terms of an integer data type of multiple nanoseconds. The second form is useful

only in environments that allow timing periods as short as nanoseconds.

**Question 181 (ID: # 17110)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Threads |
| **Description** | Exception Thrown by Sleep() |

When the `sleep()` method is used in the `Thread` class, which exception should be caught?

| | |
|---|---|
| A. | SecurityException |
| B. | IllegalMonitorStateException |
| C. | InterruptedException |
| D. | none of the above |
| E. | It doesn't throw any checked exception. |

| | |
|---|---|
| Correct Answer | C |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The sleep() method is used to temporarily stop a particular thread from functioning for a specified period of time. Meanwhile, other thread functions may interrupt the "sleeping" thread, which will throw an InterruptedException to signify that its state is changing. |

**Question 182 (ID: # 17111)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Threads |
| **Description** | Thread Methods |

Which of the following statements are true?

| | |
|---|---|
| A. | The interrupt() method of a thread makes it throw an InterruptedException when it is in turn for execution. |
| B. | The yield() method puts a thread in the waiting state. |
| C. | The sleep() method puts a thread in the ready state. |
| D. | The join() method waits for this thread to die. |
| E. | The resume() method is the recommended way to change a thread's state to runnable . |

| | |
|---|---|
| Correct Answer | A ;D ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The interrupt() method of a thread makes it throw an InterruptedException , and the join() method waits for this thread to die. resume() is deprecated, yield() causes the currently executing thread object to temporarily pause and allow other threads to execute, and the sleep() method tries to put the current thread to sleep for the specified number of milliseconds (nanoseconds). |

**Question 183 (ID: # 17112)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Threads |
| **Description** | Re-start()-ing a Thread |

What will the result be when the code below is run?

```
class MyThread extends Thread {
      public static void main(String[] args) {
            MyThread t = new MyThread();
            t.start();
            System.out.println(" one ");
            t.start();
            System.out.println(" two ");
      }
      public void run(int i) {
            System.out.println("running");
      }
}
```

A. It will print "one running two running".

B. It will print "one running" and then an exception will be thrown.

C. It will print "one two".

D. It will print "one" and then an exception will be thrown.

E. The code will not compile.

| | |
|---|---|
| Correct Answer | D |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | The run() method from Thread is not overridden by the custom run() ; it is just overloaded. Therefore, the custom run(int i) it is not executed at all. When it starts a second time, a thread will throw an IllegalThreadStateException , even if the thread has finished its job. |

**Question 184 (ID: # 17113)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | Threads |
| **Description** | Thread Constructor |

What will the output be when the code below is run?

```
1. public class Test extends Thread {
2.       public static void main(String[] args)
3.       throws InterruptedException {
4.             Test t = new Test();
5.             Thread th = new Thread(t);
6.             th.start();
7.             th.join();
8.       }
9.       public void run() {
10.            for(int i=0;i<3;i++) {
11.                  System.out.println(i + "..");
12.            }
13.      }
14. }
```

A. It will fail to compile at Line 3.

B. It will print "0..1..2..".

C. It will fail to compile at Line 5.

D. It will fail to compile at Line 7.

E. It will fail to compile at Line 11.

Correct Answer B

User Answer

Elapsed Time (seconds) 0

Explanation The program will compile and run fine, printing "0..1..2..". The Test class extends Thread , and therefore implements Runnable . Therefore, it is perfectly legal to pass it to a Thread instance constructor. The join() method will wait for this (main) thread to die (which will happen right away) and then print the text.

**Question 185 (ID: # 17114)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Threads

**Description** Synchronizing Methods

Suppose you have a class with two private members, `a` and `b`. Assuming that no other methods are declared allowing concurrent access, which of the following `getters` and `setters` declaration statements will prevent concurrent access to these members?

A. public int read() {return a+b;} public void set(int a,int b) {this.a=a;this.b=b;}

B. public synchronized int read() {return a+b;} public synchronized void set(int a,int b) {this.a=a;this.b =b;}

C. public int read() {synchronized(a) {return a+b;}} public void set(int a,int b) {synchronized(a) {this.a =a;this.b=b;}}

D. public int read() {synchronized(a) {return a+b;}} public void set(int a,int b) {synchronized(b) {this. a=a;this.b=b;}}

E. public void set(int a,int b) {synchronized(this) {this.a=a;this.b=b;}}

Correct Answer B ;E ;

User Answer

Elapsed Time (seconds) 0

Explanation Option A is totally unsynchronized. Options C and D are incorrect, due to the fact that only Objects can be synchronized. Options B and E are the correct forms of synchronization that will prevent unwanted concurrent access.

**Question 186 (ID: # 17115)**

**Subject** Java 2 Standard Edition 1.5

**Subtopic** Threads

**Description** Pausing Guaranteed

Which of the following code fragments guarantee that a thread will leave the running state?

A. wait()

B. notifyAll

| | |
|---|---|
| C. | yield() |
| D. | sleep(2003) |
| E. | activeThread.join() |
| Correct Answer | A ;D ;E ; |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | sleep() will always pause the current thread for at least the specified amount of seconds. join() will block it until the joined thread is dead. wait() always causes the current thread to go in the object's wait pool. |

**Question 187 (ID: # 152239)**

| | |
|---|---|
| **Subject** | Java 2 Standard Edition 1.5 |
| **Subtopic** | XML Processing |
| **Description** | ErrorListener |

If an application does not register a(n) _____, errors are reported to the `System.err`.

| | |
|---|---|
| A. | SourceLocator |
| B. | ErrorListener |
| C. | Source |
| D. | Result |
| E. | URIResolver |
| Correct Answer | B |
| User Answer | |
| Elapsed Time (seconds) | 0 |
| Explanation | Option B is correct. For transformation errors, a Transformer has to use this interface, rather than throwing an exception. It is up to the application to choose whether to throw an exception for different types of errors and warnings. The transformer is not required to carry on with the transformation after a call to a fatalError . |