Cs520 Quiz Reviews

by MikeBegley, Mar. 2017 | Subjects: Java | [?]

Flashcards | Memorize | Test | Games

How to study your flashcards.

Right/Left arrow keys: Navigate between flashcards.

Up/Down arrow keys: Flip the card between the front and back.

H key: Show hint (3rd side).

A key: Read text to speech.

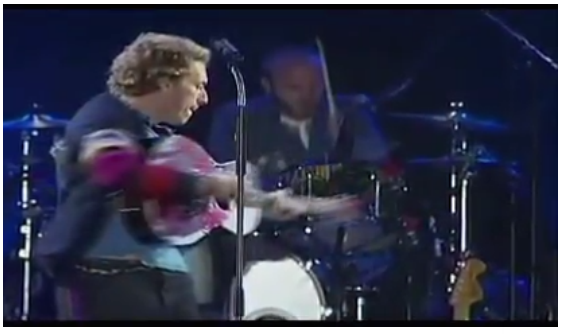
Click or Press Spacebar to Begin »

Related Flashcards

Java Review
Java quiz
Java Review Questions
SCJP 5 review

CARD RANGE TO STUDY

1 through 96



96 Cards in this Set Share Print Export Clone

Front	Back
	11.

Given the following declarations:
int a = 10;
int b = 20;
the expression.(++b - a++) will evaluate to
	++b will evaluate first to 21. The value of a is then subtracted from 21, resulting in 11. (Note that a++ is post increment operator, so it is not incremented for the expression. Where as ++b is a pre increment operator and it is incremented for the expression.)

Consider the following segment of code:
int length = 10; int sum = 0;
// insert a definition of idx here
while (idx <= length)
{
sum += idx;
idx++;
}
What is the minimum value of idx to be used before the loop so the sum would not change inside the loop?
	int idx = 11;

Which of the following loop statements always have their body executed at least once? -The while loop -Correct The do-while loop -The for loop -None of the choices	The do-while loop
Given: Which of the following, inserted at line 4, will prevent line 6 (y++;) from executing? -int y = x; -int y = 10; -int y = 11; -None of the given options	None of the given options

Within the following code:
int balance = 11;
while(balance-- >= 1)
{
if(balance > 9)
continue;
balance -= 9;
}
How many times is "continue" executed?
	1

Which of the following is the correct sequence of declaring the main() method of a Java application? public static void main(String[] args) public void static main(String[] args) static public void main(String[] args) public void static main()	public static void main(String[] args)

Given the following declarations:
int age = 30;
double salary = 25000 ;
which one of the following is a valid assignment statement?
age = salary / 100;
salary = 100 * age;
age = salary - 10000;
age = salary * age;
	salary = 100 * age;

The System.out.println("hello") method:	prints its argument, and then prints an end-of-line

Given the following declarations:
int a = 10;
int b = 30;
int c = 20;
the expression.(a + b / c + a) will evaluate to
	21

Which of the following is not a keyword of Java? class public keyword void	keyword
Which one of the following lines is not a properly formatted comment? a. /* this is a comment */ b. // This is a comment c. /* This is a comment */ d. // this is a comment //	d. // this is a comment //
What is the relationship between .java and .class files?	A .java file (with no syntax errors) can be compiled into a .class file.
In Java, a statement block is delimited by:	{ }
Java compiles the source code into an intermediate language called byte-code. What is byte-code?	It's the machine language for the Java Virtual Machine
Which one of the following is a correct assignment statement for the variable x? int x = 20,	

double x = 20;
int 20 = x;
double 20.0 = x;
	double x = 20;

Which one of the following is true about Java? a. It is not case sensitive. b. It can be compiled using the C++ compiler. c. It is one type of 4th generation language such as SQL. d. It is object oriented and is widely used in web applications.	d. It is object oriented and is widely used in web applications.
A variable whose meaning is confined to an object of a class is called	instance variable
When you want the parameters in a method to be the same as the instance variables, you use the _____ keyword.	this
The modifier private means that an instance variable can be accessed by name outside of the class definition	False

Analyze the following code:
class Test {
public static void main(String[] args) {
System.out.println(xMethod((double)5));
}
public static int xMethod(int n) {
System.out.println("int");
return n;
}
public static long xMethod(long n) {
System.out.println("long");
return n;
}
}
	The program does not compile.

What is the output of this code?	The code compiles and executes fine, and the output is "Hello, Dear."
Which of the following statements is true about the static modifier? -A static variable cannot change its value. -A static method is often written in a non-Java language and exists outside of JVM. -A static method can access instance variables -A static variable is shared by all the instances of	A static variable is shared by all the instances of the class

the class	
Which of the following does not overload the method -public int myMethod(int i, double a) { } -public double myMethod(double b, int j){ } -public int myMethod(double a, double b, int i){ } -public int myMethod(int a, int i) { }	public double myMethod(double b, int j){ }

Consider the following code:
 	
class Test
{
Test(){...}
Test(int x){...}
void method(){...}
}
 	
Which of the following code fragments, using the class defined above, would generate a compile error? The ellipses {...} are used here to indicate the presence of some code.	

What is the difference between static variable and instance variable?	static variable is shared by all the instances of the class, while instance variable is not
Which of the following statements are correct about constructor? Three choices are applicable. - Constructor can ONLY be called when a new instance of the class has to be created. - Constructor can have multiple overloaded versions. -If you do not declare a constructor, then the class has no constructor; therefore can not be instantiated. - It always has the same name as the name of the class.	Constructor can ONLY be called when a new instance of the class has to be created. Constructor can have multiple overloaded versions. It always has the same name as the name of the class.
A class can be defined by extending another class. It is called _____ .	inheritance
	The default constructor of A is invoked The constructor of B is invoked
The Object class and the Integer class both have toString() methods. Note that the Object class is the root class of Java hierarchy.	When x.toString() is invoked, the toString() method in the Integer class is used
	Animal

What output is generated when Test is run?	Animal Horse Donkey
A method that performs some actions but does not return a value can be declared a _____ method.	void
Which of the following is **NOT** correct about declaring a class? -The body of the class is bracketed by {} -The name of the class must be after the "class" keyword -It can contain instance variables and instance methods -It must be declared with at least one instance variable	It must be declared with at least one instance variable
A variable whose meaning is confined to a method definition is called an/a	local variable
Tokenizing the string "a;b;c;d" using the default delimiter results in	1 token

Consider the following code fragment:
public class ExceptionHandlingNote{
public static void main(String[] args){
String x;
System.out.println(x);
}
}
Which of the following is true:
 	
It prints out 0
It prints out null
It prints out nothing
None of the choices is true
	None of the choices is true

Which of the following are the CORRECT ways of declaring a string? String s = new String("Hello"); String s = 'hello'; char[] s = "Hello"; String s = 123;	String s = new String("Hello");
If an object must be written to and read back from a file in its entirety, the class definition of that object must	implement the Serializable interface
Consider the following line of code: String str = new String("Hello");	

Which of the following lines of code modify the string to which str refers? -str.substring(2); -str.replace('H', 'M'); -str.trim(); -None of the choices.	None of the choices.
The StringTokenizer class is used for	examining the individual tokens in the given string based on the specified delimiters
Which of the following statement is correct about Exception? -Exception is not a class. -An exception must be caught before being thrown. -An exception can be either handled or be declared as thrown. -try{} block must always be used to capture an exception.	An exception can be either handled or be declared as thrown.
Which one of the following classes from the java.io package can be used for writing objects to a file? -ObjectOutputStream -BufferedReader -FileReader -File	ObjectOutputStream
Which of the following is correct about the String class? -The replace() method will modify the string object which is calling the method. -The trim() method will remove the white space at the end of the current string object. -substring() method will return a new string object, instead of modifying the current string object. -String class is equivalent to string class.	substring() method will return a new string object, instead of modifying the current string object.
You can not place a try block and its following catch blocks inside a larger try block or inside a larger catch block.	False
The catch clause has _____ parameter(s).	one
Ye code new StringBuffer("abcd").reverse().substring(1, 2) results in the following string:	"c"
	y
The method trim() of the String class trims off:	Leading and trailing white spaces

The method trim() of the String class trims off	Leading and trailing white spaces
Tokenizing the string "a,b;c,d;e,f" using the comma delimiter results in	4 tokens
Suppose String s= "Java"; which of the following will change s to "java". More than one choice is valid. -s.toLowerCase(); -s.toUpperCase(); -s=s.toLowerCase(); -s=s.replace('J','j');	s=s.toLowerCase(); s=s.replace('J','j');
Which one of the following classes from the java.io package can be used for writing text data to a file? -PrintWriter -BufferedReader -FileReader -File	PrintWriter
Suppose s1 and s2 are two non-empty strings with the length of s1 greater than the length of s2. Which of the following statements or expressions will generate a compile error? -String s3 = s1 + s2; -char c = s1.substring(0,1); -s2 = s1.replace('s', 's'); -s1 = s2.substring(s1.length());	char c = s1.substring(0,1);

What is true about the following partial code?
if("String".toString() == "String")
System.out.println("Equal");
else
System.out.println("Not Equal");
	The code will compile an print "Equal".

Tokenizing the string "a,b;c,d;e,f" using the semi-colon delimiter results in	3 tokens
Which of the following statements is true about exception handling? -Exception is not a type of class -When you have a try{} clause, you must use catch as well -Some exceptions do not require the programmer to handle using try-catch -When a program would possibly throw an exception, this exception must be caught using a try clause	Some exceptions do not require the programmer to handle using try-catch
A _____ is the unit of space on the screen that Java uses for graphics.	pixel

Consider the code button.addActionListener(this); which of the following is **NOT** correct? -The "this" referred to in the code must be an implementation of one or more listeners. -The code will register the button with a listener. -The listener that "this" referrs to should have the method actionPerformed(ActionEvent e) implemented. -One must declare the button to implement one or more Listeners.	One must declare the button to implement one or more Listeners.
The _____ is an object that waits for an event to occur and responds in some way when it does.	Listener
Which of the following draws a circle whose center position is (400, 400) and whose radius is 50? -g.drawCircle(350, 350, 100, 100); -g.drawOval(350, 350, 100, 100); -g.drawCircle(400, 400, 50, 50); -g.drawOval(400, 400, 50, 50);	g.drawOval(350, 350, 100, 100);
A button control should have a registered _____ associated with it to take some action when the user clicks on it.	listener
In event-driven programming, a Java program includes one or several of the following steps: 1. create a component object and add it to a container 2. specify which class implements a component's event listener 3. implement an event handler method and specify what program method calls it 4. register the component as a listener for the application frame Which steps actually have to be included?	(1) and (2)

Given the sequence of the following statements,
g.drawLine(50,100,150,100);
g.drawLine(50,50,150,50);
g.drawLine(150,50,150,100);
g.drawLine(50,50,50,100);
which of the following can replace the above sequence?
-g.drawRect(150,50,100,50);
-g.drawRect(50,100,100,50);
-g.drawRect(50,50,100,50);
-g.drawRect(100,100,50,50);
	g.drawRect(50,50,100,50);

Suppose an ArrayList of String objects, myList, has the strings "1", "2", and "3" added to it in that order.
The code
ListIterator it = myList.listIterator();
while (it.hasNext()) {
System.out.print(it.next());
it.previous();
}
results in the following output:
	An infinite number of 1's

Suppose a HashSet of String objects, mySet, has the strings "1", "2", "3", "4", and "5" added to it in that order. When iterating over the elements in this set, the output	order of the elements cannot be guaranteed.
Suppose a HashSet of String objects, mySet, has the strings "1", "2", "3", "4", and "5" added to it in that order. The same set of strings is added one more time. The size of the mySet will be	5
Suppose we have an array of String objects (with at least one element) identified by the variable 'names'. Which of the following loops will CORRECTLY process each element in the array? -for (int i = 0; i < names.length; i++) ... -for (int i = 0; i < names.length(); i++) ... -int i = 0; while (i <= names.length) {.... i++; } -None of the above	for (int i = 0; i < names.length; i++) ...
Assume the signature of the method xMethod is as follows: public static void xMethod(double[] a) Which of the following could be used to invoke xMethod(double[] a)? -xMethod(5); -xMethod({3, 4}); -xMethod(new int[2]); -xMethod(new double[2]);	xMethod(new double[2]);
Consider the following array declaration inside the body of a method. int[ ] values = new int[10]; Accessing the array element, values[0], right after the above declaration results in:	the value 0
Which of the following initializer correctly initializes the elements of an array named myDoubles? -double myDoubles[double] = {0.0, 1.0, 1.5, 2.0, 2.5}; -double myDoubles[5] = new double (0.0, 1.0, 1.5, 2.0, 2.5); -double [ ] myDoubles = {0.0, 1.0, 1.5, 2.0, 2.5};	double [ ] myDoubles = {0.0, 1.0, 1.5, 2.0, 2.5};

-array myDoubles[double] = {0.0, 1.0, 1.5, 2.0, 2.5};	
Consider the following array declaration inside the body of a method. String[ ] values = new String[10]; Invoking the length method on the array element, values[0], right after the above declaration results in:	A compilation error since there is no initial value assigned to it
Consider the following array declaration inside the body of a method. int[ ][ ] values = {{1},{2,3},{4,5,6}}; The length of the values array is:	3
Consider the following array declaration inside the body of a method. int[ ][ ] values = {{1},{2,3},{4,5,6}}; The expression (values[0] + values[1]) results in:	a syntax or compilation error
Consider the following array declaration inside the body of a method. int[ ][ ] values = {{1},{2,3},{4,5,6}}; The expression.(values[1][1] + values[2][2]) results in:	9
Consider the following ArrayList declaration inside the body of a method. ArrayList myList = new ArrayList(); Accessing the element, myList.get(0), right after the above declaration results in:	An exception that will be thrown at runtime.
Consider the following ArrayList declaration inside the body of a method. ArrayList myList = new ArrayList(); Which of the following statements are true before the above ArrayList is modified? More than one choice may be applicable.	The capacity of the above ArrayList is 10 The size of the above ArrayList is 0 The capacity of the above ArrayList is 10
Consider the following ArrayList declaration and its iterator inside the body of a method. ArrayList myList = new ArrayList(); Iterator it = myList.iterator(); Invoking the it.next() method right after the above declaration results in:	An exception that will be thrown at runtime.

Suppose an ArrayList of String objects, myList, has the strings "1", "2", and "3" added to it in that order. The code Iterator it = myList.iterator(); while (it.hasNext()) System.out.print(it.next()); results in the following output:	123
Assume that a LinkedList is used for a queue implementation and has the strings "1", "2", "3", "4", and "5" added to it in that order. The peek method results in the string	"1"
Assume that a LinkedList is used for a queue implementation and has the strings "1", "2", "3", "4", and "5" added to it in that order. After invoking theremove, the peek method results in the string	"2"

Given an empty ArrayDeque of String objects, myDeck, the strings "1", "2", "3", and "4" are added to it using the addFirst and addLast methods alternatively as shown below.
myDeck.addFirst("1");
myDeck.addLast("2");
myDeck.addFirst("3");
myDeck.addLast("4");
The expression myDeck.removeFirst() + myDeck.removeLast() results in
	34

Given an empty ArrayDeque of String objects, myDeck, the strings "1", "2", "3", and "4" are added to it using the addFirst and addLast methods alternatively as shown below.
myDeck.addFirst("1");
myDeck.addLast("2");
myDeck.addFirst("3");
myDeck.addLast("4");
After invoking the removeFirst and removeLast methods, the peek method results in
	1

Given an empty LinkedList of Integer objects, myList, the values 1, 2, 3, 4, and 5 are added to it using the addFirst method.
myList.addFirst(1);
myList.addFirst(2);
myList.addFirst(3);
myList.addFirst(4);
myList.addFirst(5);
The expression myList.get(1) + myList.get(2) results in
	7

Given an empty LinkedList of Integer objects, myList, the values 1, 2, 3, 4, and 5 are added to it using the addLast method.
myList.addLast(1);
myList.addLast(2);
myList.addLast(3);
	5

myList.addLast(4);
myList.addLast(5);
The expression myList.get(1) + myList.get(2) results in	

Given an empty LinkedList of Integer objects, myList, the values 1, 2, 3, 4, and 5 are added to it using the addLast method.
myList.addLast(1);
myList.addLast(2);
myList.addLast(3);
myList.addLast(4);
myList.addLast(5);
The expression myList.contains("2") returns true.
	False

Suppose a Stack of String objects, myStack, has the strings "1", "2", "3", "4", and "5" pushed onto it in that order. The myStack.pop() method results in the string	"5"
Suppose a Stack of String objects, myStack, has the strings "1", "2", "3", "4", and "5" pushed onto it in that order. After invoking the peek() method, the pop()method results in the string	"5"
Suppose a Stack of String objects, myStack, has the strings "1", "2", "3", "4", and "5" pushed onto it in that order. After invoking the pop() method, the peek()method results in the string	"4"

Consider the following class definition.
public class Employee {
private int salary = 50000;
public void payRaise() {
salary += 500;
}
public void payCut() {
salary -= 500;
}
}
 	
Suppose that there are two threads (the Supervisor thread and the CEO thread) with access to the same Employee object. One thread attempts to give thepayRaise while the other thread attempts a payCut concurrently. What would be the result?
	The employee's salary will either be 49500, 50000, or 50500.

Consider the following class definition.
public class TestThread implements Runnable{
public void run() {
System.out.println("Hello");
}
}
The code invoking the start() method on a Thread
	The output "Hello"

object created using the TestThread object
new Thread(new TestThread()).start();

Which one of the following statements causes the current thread to sleep for a random amount of time up to a maximum of one minute.
-Thread.sleep(60000 * Math.random());
-Thread.sleep((long) (60000 * Math.random()));
-Thread. .sleep((long) (10000 * Math.random()));
-Thread.sleep(10000 * Math.random());

Thread.sleep((long) (60000 * Math.random()));

Consider the following class definition.
public class Employee {
private int salary = 50000;
public synchronized void payRaise() {
salary += 500;
}
public synchronized void payCut() {

salary -= 500;
}
}
Suppose that there are two threads (the Supervisor thread and the CEO thread) with access to the same Employee object. One thread attempts to give thepayRaise while the other thread attempts a payCut concurrently. What would be the result?

The employee's salary will always be 50000.

Consider the following class definition.
public class TestThread {
public void run() {
System.out.println("Hello");
}
}
The code invoking the start() method on a Thread object created using the TestThread object
new Thread(new TestThread()).start();
results in:

A syntax or compile time error

Consider the following class definition.
class Test {
public synchronized void foo()
throws InterruptedException {
wait();
System.out.println("Goodbye");
}
public synchronized void bar()
throws InterruptedException {
wait();
System.out.println("Hello");
}
}

Suppose two threads running concurrently share the same Test object. One thread invokes the bar method and the other thread invokes the foo method of the same object. What would be the result?

The two threads are deadlocked and wait forever until interrupted.

Consider the following class definition.
public class TestThread {
public void run() {
System.out.println("Hello");
}
}
The code invoking the start() method on a TestThread object results in:
	A runtime exception

Consider the following class definition.
public class TestThread extends Thread{
public void start() {
System.out.println("Goodbye");
}
public void run() {
System.out.println("Hello");
}
}
 	
The code invoking the start() method on a TestThread object results in:
	The output "Goodbye"

Consider the following class definition.
class Test {
public synchronized void foo() {
bar();
System.out.println("Goodbye");
}
public synchronized void bar() {
System.out.println("Hello");
}
}
 	
When a thread invokes the foo() method on a Test object, (when no other thread is sharing this object), what would be the result?
	The output will be "Hello" followed by "Goodbye"

Consider the following class definition.
class Test {
public synchronized void foo() throws InterruptedException {
wait();
System.out.println("Goodbye");
}
public synchronized void bar() {
System.out.println("Hello");
notify();
}
}
 	
Suppose two threads share the same Test object. One thread always invokes the bar method first before the other thread invokes the foo method of the same object. What would be the result?
	The output will be "Hello" and the other thread waits forever until interrupted.