lgsi('geekathon');

ANSWER GUIDE

JAVA TEST – 10 May 2012

1. What is the prototype of the default constructor?

A. Test( )

B. Test(void)

C. public Test( )

D. public Test(void)

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.

Read More: Declarations & Access Control

2. What is the narrowest valid returnType for methodA in line 3?

```
public class ReturnIt
{
    returnType methodA(byte x, double y) /* Line 3 */
    {
        return (long)x / y * 2;
    }
}
```

A. int

B. byte

C. long

D. double

Explanation:

However A, B and C are all wrong. Each of these would result in a narrowing conversion. Whereas we want a widening conversion, therefore the only correct answer is D. Don't be put off by the `long` cast, this applies only to the variable x and not the rest of the expression. It is the variable y (of type `double`) that forces the widening conversion to`double`.

Java's widening conversions are:

- From a byte to a short, an int, a long, a float, or a double.

- From a short, an int, a long, a float, or a double.

- From a char to an int, a long, a float, or a double.

- From an int to a long, a float, or a double.

- From a long to a float, or a double.

- From a float to a double.

Read More: Declarations & Access Control

3. Which two cause a compiler error?

1. float[ ] f = new float(3);
2. float f2[ ] = new float[ ];
3. float[ ]f1 = new float[3];
4. float f3[ ] = new float[3];
5. float f5[ ] = {1.0f, 2.0f, 2.0f};

A. 2, 4

B. 3, 5

C. 4, 5

D. 1, 2.

Explanation:

(1) causes two compiler errors ('[' expected and illegal start of expression) because the wrong type of bracket is used, `( )` instead of `[ ]`. The following is the correct syntax:`float[ ] f = new float[3];`

(2) causes a compiler error ('{' expected) because the array constructor does not specify the number of elements in the array. The following is the correct syntax: `float f2[ ] = new float[3];`

(3), (4), and (5) compile without error.

Read More: Declarations & Access Control

4. What will be the output of the program?

```
public class Test
{
    public static void main(String args[])
    {
        class Foo
        {
            public int i = 3;
        }
        Object o = (Object)new Foo();
        Foo foo = (Foo)o;
        System.out.println("i = " + foo.i);
    }
}
```

A. i = 3

B. Compilation fails.

C. i = 5

D. A `ClassCastException` will occur.

Explanation:

NA

Read More: Declarations & Access Control

5. What will be the output of the program?

```
public class A
{
    void A() /* Line 3 */
    {
        System.out.println("Class A");
    }
    public static void main(String[] args)
    {
        new A();
    }
}
```

A. Class A

B. Compilation fails.

C. An exception is thrown at line 3.

D. The code executes with no output.

Explanation:

Option D is correct. The specification at line 3 is for a method and not a constructor and this method is never called therefore there is no output. The constructor that is called is the default constructor.

Read More: Declarations & Access Control

6. 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 3

B. 8 2

C. 8 3

D. 8 5

Explanation:

The first two iterations of the `for` loop both `x` and `y` are incremented. On the third iteration `x` is incremented, and for the first time becomes greater than 2. The short circuit or operator `||` keeps `y` from ever being incremented again and `x` is incremented twice on each of the last three iterations.

Read More: Operators & Assignments

7. Which three statements are true?

```
import java.awt.Button;
class CompareReference
{
    public static void main(String [] args)
    {
        float f = 42.0f;
        float [] f1 = new float[2];
        float [] f2 = new float[2];
        float [] f3 = f1;
        long x = 42;
        f1[0] = 42.0f;
    }
}
```

1. f1 == f2
2. f1 == f3
3. f2 == f1[1]

4. x == f1[0]
5. f == f1[0]

A. 1, 2 and 3

B. 2, 4 and 5

C. 3, 4 and 5

D. 1, 4 and 5

Explanation:

(2) is correct because the reference variables `f1` and `f3` refer to the same array object.

(4) is correct because it is legal to compare integer and floating-point types.

(5) is correct because it is legal to compare a variable with an array element.

(3) is incorrect because `f2` is an array object and `f1[1]` is an array element.

Read More: Operators & Assignments

8. What will be the output of the program?

```
public class MyProgram
{
    public static void main(String args[])
    {
        try
        {
            System.out.print("Hello world ");
        }
        finally
        {
            System.out.println("Finally executing ");
        }
    }
}
```

A. Nothing. The program will not compile because no exceptions are specified.

B. Nothing. The program will not compile because no catch clauses are specified.

C. Hello world.

D. Hello world Finally executing

Explanation:

Finally clauses are always executed. The program will first execute the `try` block, printing Hello world, and will then execute the finally block, printing Finally executing.

Option A, B, and C are incorrect based on the program logic described above. Remember that either a catch or a finally statement must follow a try. Since the finally is present, the catch is not required.

Read More: Exceptions

9. Which statement is most true concerning this code?

```
System.out.print("Start ");
try
{
    System.out.print("Hello world");
    throw new FileNotFoundException();
}
System.out.print(" Catch Here "); /* Line 7 */
catch(EOFException e)
{
    System.out.print("End of file exception");
}
catch(FileNotFoundException e)
{
    System.out.print("File not found");
}
```

and given that `EOFException` and `FileNotFoundException` are both subclasses of`IOException`, and further assuming this block of code is placed into a class.

A. The code will not compile.

B. Code output: Start Hello world File Not Found.

C. Code output: Start Hello world End of file exception.

D. Code output: Start Hello world Catch Here File not found.

Explanation:

Line 7 will cause a compiler error. The only legal statements after `try` blocks are either`catch` or finally statements.

Option B, C, and D are incorrect based on the program logic described above. If line 7 was removed, the code would compile and the correct answer would be Option B.

Read More: Exception

10. Which class does not override the equals() and hashCode() methods, inheriting them directly from class Object?

A. java.lang.String

B. java.lang.Double

C. java.lang.StringBuffer

D. java.lang.Character

Explanation:

`java.lang.StringBuffer` is the only class in the list that uses the default methods provided by class `Object`.

Read More: Objects & Collections

11. 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

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 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.

Read More: Objects & Collections

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

A. No statement required.

B. import java.io.*;

C. include java.io.*;

D. import java.io.PrintWriter;

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.

Read More: Objects & Collections

13. Which is true about an anonymous inner class?

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

B. It can extend exactly one class and can implement multiple interfaces.

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

D. It can implement multiple interfaces regardless of whether it also extends a class.

Explanation:

Option C is correct because the syntax of an anonymous inner class allows for only one named type after the new, and that type must be either a single interface (in which case the anonymous class implements that one interface) or a single class (in which case the anonymous class extends that one class).

Option A, B, D, and E are all incorrect because they don't follow the syntax rules described in the response for answer Option C.

Read More: Inner Classes

14. Which statement, if placed in a class other than MyOuter or MyInner, instantiates an instance of the nested class?

```
public class MyOuter
{
    public static class MyInner
    {
        public static void foo() { }
    }
}
```

A. MyOuter.MyInner m = new MyOuter.MyInner();

B. MyOuter.MyInner mi = new MyInner();

C. MyOuter m = new MyOuter(); MyOuter.MyInner mi = m.new MyOuter.MyInner();

D. MyInner mi = new MyOuter.MyInner();

Explanation:

`MyInner` is a static nested class, so it must be instantiated using the fully-scoped name of `MyOuter.MyInner`.

Option B is incorrect because it doesn't use the enclosing name in the new.

Option C is incorrect because it uses incorrect syntax. When you instantiate a nested class by invoking new on an instance of the enclosing class, you do not use the enclosing name. The difference between Option A and C is that Option C is calling new on an instance of the enclosing class rather than just new by itself.

Option D is incorrect because it doesn't use the enclosing class name in the variable declaration.

Read More: Inner Classes

15. Which two are valid constructors for Thread?

1. Thread(Runnable r, String name)
2. Thread()
3. Thread(int priority)
4. Thread(Runnable r, ThreadGroup g)
5. Thread(Runnable r, int priority)

A. 1 and 3

B. 2 and 4

C. 1 and 2

D. 2 and 5

Explanation:

(1) and (2) are both valid constructors for `Thread`.

(3), (4), and (5) are not legal `Thread` constructors, although (4) is close. If you reverse the arguments in (4), you'd have a valid constructor.

Read More: Threads

16. 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

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.

Read More: Threads

17. Under which conditions will a currently executing thread stop?

1. When an interrupted exception occurs.
2. When a thread of higher priority is ready (becomes runnable).
3. When the thread creates a new thread.
4. When the `stop()` method is called.

A. 1 and 3

B. 2 and 4

C. 1 and 4

D. 2 and 3

Explanation:

The statements (2) and (4) makes currently executing thread to stop.

Read More: Threads

18. Which statement is true?

A. The `notifyAll()` method must be called from a synchronized context.

B. To call `wait()`, an object must own the lock on the thread.

C. The `notify()` method is defined in class java.lang.Thread.

D. The `notify()` method causes a thread to immediately release its locks.

Explanation:

Option A is correct because the `notifyAll()` method (along with `wait()` and`notify()`) must always be called from within a synchronized context.

Option B is incorrect because to call `wait()`, the thread must own the lock on the object that `wait()` is being invoked on, not the other way around.

Option C is wrong because `notify()` is defined in `java.lang.Object`.

Option D is wrong because `notify()` will not cause a thread to release its locks. The thread can only release its locks by exiting the synchronized code.

Read More: Threads

19. When is the Float object, created in line 3, eligible for garbage collection?

```
public Object m()
{
    Object o = new Float(3.14F);
    Object [] oa = new Object[1];
    oa[0] = o; /* Line 5 */
    o = null;  /* Line 6 */
    oa[0] = null; /* Line 7 */
    return o; /* Line 8 */
}
```

A. just after line 5

B. just after line 6

C. just after line 7

D. just after line 8

Explanation:

Option A is wrong. This simply copies the object reference into the array.

Option B is wrong. The reference o is set to null, but, `oa[0]` still maintains the reference to the `Float` object.

Option C is correct. The thread of execution will then not have access to the object.

Read More: Garbage Collections

20. What will be the output of the program?

```
String a = "ABCD";
String b = a.toLowerCase();
b.replace('a','d');
b.replace('b','c');
System.out.println(b);
```

A. abcd

B. ABCD

C. dccd

D. dcba

Explanation:

`String` objects are immutable, they cannot be changed, in this case we are talking about the replace method which returns a new `String` object resulting from replacing all occurrences of `oldChar` in this string with `newChar`.

```
b.replace(char oldChar, char newChar);
```

But since this is only a temporary `String` it must either be put to use straight away i.e.

```
System.out.println(b.replace('a','d'));
```

Or a new variable must be assigned its value i.e.

```
String c = b.replace('a','d');
```

Read More: Java.lang Class

REMEMBER: PREPARATION IS KEY