Boot camp 2015. Java Exam. #101

1. Java uses call by value. What is the value that is being passed into routine by the following method call?

```
double a[] = { 1.2, 3.4, 5.6 };
routine(a);
```

a) A copy of the array
b) Values of the array elements
c) A reference to the array object
d) None of the above

2. Which one Collection class allows you to grow or shrink its size and provide indexed access to its elements?

a) `java.util.ArrayList`
b) `java.util.HashMap`
c) `java.util.TreeSet`
d) `java.util.Hashtable`

3. What will be the output of the following code?

```
Object o = null;
System.out.println(o.equals(null));
```

a) true
b) false
c) Runtime error
d) Undefined

4. What exception will be thrown from the following block of code?

```
try {
    throw new TryException();
} catch (Exception e) {
    throw new CatchException();
} finally {
    throw new FinallyException();
}
```

a) TryException
b) CatchException
c) FinallyException
d) None of the above

5. What will be the result of this code:

```
String a = "abc";
String b = a;
b += "def";
```

a) a == "abcdef", b == "abcdef", a == b
b) a == "abcdef", b == "abcdef", a != b
c) a == "abc", b == "abcdef", a != b
d) compilation error

6. Given:
- A and E are classes
- B and D are interfaces
- C is an abstract class

a) `class F implements B, D { }`
b) `class F extends A, E { }`
c) `class F extends A implements C { }`
d) `class F implements B, C { }`

7. What will be the output of the following code?

```
public static void main(String[] args) {
    int a[] = { 1, 2, 3, 4 };
    System.out.println(a instanceof Object);
}
```

a) true
b) false
c) Compilation error
d) 1 2 3 4

8. Range of values for **int** primitive type is
a) $-2^{15} .. 2^{15}-1$
b) $-2^{31} .. 2^{31}-1$
c) $-2^{63} .. 2^{63}-1$
d) None of the above

9. What will be the output of the following code?

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

a) x = 1
b) x = 3
c) Compilation fails
d) No output

10. Two or more methods with the same name in the same class, but with different list of arguments is called
a) Overloading
b) Overriding
c) Abstraction
d) Synchronization

11. Which OOP principle restricts access to object's state by directly changing field values?
a) Abstraction
b) Inheritance
c) Encapsulation
d) Polymorphism

12. Static variables ...
a) ... can be accessed only through a class instance
b) ... share the same value between all instances of a class
c) ... are not visible from other classes
d) ... cannot be disposed by Garbage Collector

13. What is the name of the abstract base class for data streams dealing with byte (non-character) input?
a) `InputStream`
b) `OutputStream`
c) `Scanner`
d) `BufferedReader`

14. What will be the output of the following code?

```
String a = "abc";
String b = new String("abc");
System.out.print(a.equals(b));
System.out.print(" ");
System.out.print(a==b);
```

a) true false
b) true true
c) false true
d) false false

15. Which of the operators are correct:
a) `int q = 12;`
b) `int q = 12L;`
c) `int q = (int)12L;`
d) `int q = (long)12;`

16. Which of the definitions are correct:

a) `int[][] q = new int[4][];`
b) `int q[][] = new int[4][];`
c) `int[4][5] q;`
d) `int[][] q[4][];`

17. What are the correct loops on array `int[] array = {1, 2, 3};`

a) `for (int q : array) { foo(array[q]); }`
b) `for (int q = 0; q < array.length; q++) { foo(array[q]); }`
c) `for (int q = 0; q <= array.length; q++) { foo(array[q]); }`
d) `for (int q : array) { foo(q); }`

18. Keyword "final":

a) forbids creating subclasses for marked class
b) forbids overriding marked method in subclasses
c) marks method as static
d) makes instance of the object immutable

19. Java is one of ...

a) declarative languages
b) imperative languages
c) logical languages
d) object-oriented languages

20. Each object, according to OOP paradigm, should have

a) identity
b) virtual methods
c) state
d) abstract methods

21. If you want to make field of your object accessible only from inside this class, you should use modifier:

a) private
b) protected
c) final
d) implicit

22. Which is a correct way to distinguish method parameters from object fields:

```
int param;
void foo(int param) {
     //TODO: replace this line
}
```

a) `param = this.param;`
b) `param = param;`
c) `this.param = param;`
d) `self.param = param;`

23. This code will be compiled only if `MyCollection` class implements:

```
MyCollection collection = new MyCollection();
for (Object o: collection) {
      //TODO
}
```

a) `Iterator` interface
b) `Iterable` interface
c) `Comparable` interface
d) `Sortable` interface

24. What can you add here to make this code compile:

```
double foo() {
    int a = 4;
    //replace this line
}
```

a) `return (double)a;`
b) `return a;`
c) `throw new Exception();`
d) `while (true);`

25. Copy constructor is a special constructor that...
a) takes an object as input and returns its deep copy
b) takes an object as input and returns its shallow (field-by-field) copy
c) takes an object as a parameter and allows access to its private fields
d) is a special method of an objects, that creates its copy

26. Enums in Java ...
a) are classes extending `java.lang.Enum` class
b) are classes implementing `java.util.Enumerable` interface
c) are primitive types
d) there's no such data type in Java

27. The result of this code will be

```
int value = 42;
ArrayList<Double> ald = new ArrayList<>();
ArrayList<Integer> ali = new ArrayList<>();
ald.add(new Double(value));
ali.add(value);
System.out.print(ald.getClass() == ali.getClass());
System.out.print(" ");
System.out.print(ald.get(0).equals(ali.get(0)));
```

a) true true
b) true false
c) false true
d) false false

28. Byte order mark (BOM) is
a) a bit in a byte that specifies the order (right-to-left; left-to-right) of bits in byte
b) is byte in the beginning of the file that specifies the order (right-to-left; left-to-right) of bits in byte
c) from 2 to 4 bytes in the beginning of the file that specify which Unicode representation is used in a file
d) 2 bytes in the beginning of the file that specify either we use Unicode in a file or not

29. Class `java.io.File` has a method:
a) `isDirectory();`
b) `read();`
c) `exists();`
d) `open();`

30. To read text file word-by-word it is better to use class:
a) `InputStream`
b) `FileInputSteam`
c) `BufferedReader`
d) `Scanner`

31. Method `nextInt()` of class `java.util.Scanner` for the provided file will return:
File text:
1000

a) 825241648
b) 1000
c) 1
d) 0

32. In Java to dispose an object you should:
a) call destructor method
b) use delete keyword
c) set variable to null and wait for merci from garbage collector
d) call `GC.collect(yourObject);`

33. To serialize object in Java you have to follow combination of these rules:
a) implement `Serializable` interface in your object
b) implement method `serialize()` of `Serializable` interface in your object
c) just call `yourObject.serialize()`
d) call `writeObject(yourObject) method of instance of ObjectOutputStream.`

34. Object serialization in Java implement strategy of:
a) deep copying
b) shallow copying
c) memory dumping
d) public-only copying

35. If some method throws **checked exception**, ...
a) that forces you to either write "try-catch" block or use "throws" declaration
b) you can detect this fact only in runtime
c) you can detect this fact in compile time by compilation error
d) that forces you to write throws Throwable declaration in main method.

36. When will happen to this code:

```
public class Parent {
	public void f() throws Exception {}
}
....
public class Child extends Parent {
	@Override
	public void f() throws Throwable {}
}
....
try {
	Parent ref = new Child();
	ref.f();
} catch(Exception e) {}
```

a) this code will not compile because you cannot use `Throwable` in "throws" declaration
b) this code will not compile because you cannot make "throws" scope of subclass wider, that of parent class
c) this code will compile and potential exception will be caught
d) this code will compile and potential exception will be missed

37. Try-catch-finally operator can have:
a) only one `catch (ThrowableSubclass e) { … }` declaration
b) many `catch (ThrowableSubclass e) { … }` declarations for different ThrowableSubclass types.
c) few `finally { }` declarations
d) at most one `finally { }` declaration

38. Class-wide tearDown method with annotation @AfterClass executes
a) once before all test methods of the class
b) before each test method of the class
c) once after all test methods of the class
d) after each test method of the class

39. Usual test architecture consist of the following parts:
a) Act
b) Assert
c) Arrange
d) Arrive

40. In try-catch-finally operator **finally** block is responsible for:
a) catching exceptions
b) executing code, in case exception was not thrown
c) executing code, even if exception was thrown
d) logging exceptions

---

41. What will be the output of the program:

```
public class Class3 {
	public static int value = 3;
}
...
public class Class4 extends Class3 {
	static int value = 4;

	static {
		Class3.value = 5;
	}

	{
		value = 6;
	}
	public Class4(int value) {
		value = this.value;
	}
}
...
public static void main(String[] args) {
	System.out.print(Class3.value);
	System.out.print(Class4.value);
	System.out.print(Class3.value);
	Class4 f = new Class4(7);
	System.out.print(Class4.value);
}
```

---

42. What will be the output for this code:

```
String value = "2";
switch (value) {
	case "1": System.out.print("A");
	case "2": System.out.print("B");
	case "3": System.out.print("C"); break;
	default: System.out.print("D"); break;
}
```

---

43. What will be the result/output for this code

```
Set c = new HashSet();
c.add("One");
c.add("Two");
c.add("One");
for (Object x : c)  System.out.print(x);
```

---

44. Here's the code. Write console output

```
int amount = 1000;
Integer amountBoxed = amount;
try {
	try {
		if (amountBoxed.equals(amount))
			throw new Exception("A");
		else
			throw new Exception("B");
	} catch (Exception e) {
		System.out.print(e.getMessage() + "1");
		throw e;
	} catch (Error e) { System.out.print(e.getMessage() + "2");
	} catch (Throwable e) {   System.out.print(e.getMessage() + "3");
	} finally { System.out.print("F"); }
} catch (Exception e) {
	System.out.println("!");
}
```

---

45. Given a code. What will be the output

```
public class C1 {
   public void boo() { System.out.println("boo!"); }
}
...
public abstract class C2 extends C1 {
   public abstract void foo();
   public void moo() { System.out.println("moo!"); }
}
...
public class C3 extends C2 {
   public void foo() {
	if (this instanceof C2)   boo();
	else moo();
   }
}
...
C1 c1 = new C3();
((C2)c1).foo();
```

---

46. What will be the output for this code

```
ArrayList<Integer> list = new ArrayList<>();
for (int i = 0; i < 10; i++)
	list.add(i / 2);
System.out.print(Collections.frequency(list, new Integer(4)));
System.out.print(Collections.frequency(list, 5));

int v = 0;
for (Integer o : list)
	if(new Integer(o.intValue()) == new Integer(v++ / 2))
		System.out.print(o);
```