Q1:

|  |
| --- |
| Objects that survive multiple rounds of garbage collection may become tenured |
| |  |  | | --- | --- | | Correct Answer: | True | |

Q2:

|  |
| --- |
| A JVM address is 32 bits allowing up to 8 gigabytes of memory  The size of an address in the JVM is 32 bits. The JVM can therefore address up to 4 gigabytes (2^32) of memory. |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |  |  | | Q3: |  | |
| The object reference *this*is stored as local variable 0 on the stack |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q4:

|  |
| --- |
| The *finalise*method is used to signal an object is ready for de-allocation  Finalizers get invoked when JVM figures out that this particular instance should be garbage collected. |
| |  |  | | --- | --- | | Correct Answer: | Correct False |   Q5: |
| The stack stores values of primitive type and references to objects / arrays |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q6:

|  |
| --- |
| Alternative implementations of the JVM are permitted provided they give the same results |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q7:

|  |
| --- |
| The arguments of a class method are local variables 0, 1, 2 and so on  Arguments are the variables we use in the main method, for example. Parameters we use in other methods.  <https://www.geeksforgeeks.org/argument-vs-parameter-in-java/> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q8:

|  |
| --- |
| The Java Virtual Machine (JVM) defines how to execute Java bytecode |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q9:

|  |
| --- |
| Each stack frame stores the method’s return address |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q10:

|  |
| --- |
| Each instance of the same string shares the same immutable interned string  Check mock test notes for explanation. |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q11:

|  |
| --- |
| When a method is called, the return address is pushed on the stack |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q12:

|  |
| --- |
| Each instance of the same literal string shares the same immutable interned string |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q13:

|  |
| --- |
| A JVM interpreter is a slower way of running Java bytecode |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q14:

|  |
| --- |
| Each Java bytecode is followed by 1 or more operands  **Each bytecode is composed of one byte that represents the opcode along with zero or more bytes for operands**. <https://en.wikipedia.org/wiki/Java_bytecode> |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q15:

|  |
| --- |
| An object reference is represented by a 32 bit pointer addressing the heap |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q16:

|  |
| --- |
| The JVM is a target for Kotlin, Python, Ruby, and Scala as well as Java  <https://en.wikipedia.org/wiki/List_of_JVM_languages> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q17:

|  |
| --- |
| The JVM has only one register, the program counter  The JVM has a program counter and three registers that manage the stack.  <https://www.javaworld.com/article/2077184/the-lean--mean--virtual-machine.html> |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q18:

|  |
| --- |
| Native methods, where supported, also use the native processor stack  A native method is a Java method whose implementation is written in another programming language. |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q19:

|  |
| --- |
| The Java instruction *iconst\_*0 is represented as the byte 0x60  IDK if this is a mistake. |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q20:

|  |
| --- |
| The JVM has one stack for each instance of the thread class |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q21:

|  |
| --- |
| WORA stands for Write Once, Read Anywhere  WORA – **write once, run anywhere** |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q22:

|  |
| --- |
| Each Java bytecode opcode takes 1 byte  <https://en.wikipedia.org/wiki/Java_bytecode> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q23:

|  |
| --- |
| The heap and stack are aligned on word (32 bit) boundaries |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q24:

|  |
| --- |
| Just-in-time compilers translate Java directly to native code  <https://en.wikipedia.org/wiki/Just-in-time_compilation> |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q25:

|  |
| --- |
| The JVM uses the special method name <init> identify object constructors |
| |  |  | | --- | --- | | Correct Answer: | Correct True | | Q26:   |  | | --- | | There is one stack frame for each active method  <https://alvinalexander.com/scala/fp-book/recursion-jvm-stacks-stack-frames/> | | |  |  | | --- | --- | | Correct Answer: | Correct True | | |  | |

Q27:

|  |
| --- |
| The JVM has a stack pointer in addition to the program counter  <https://www.javaworld.com/article/2077184/the-lean--mean--virtual-machine.html> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q28:

|  |
| --- |
| Each Java bytecode is followed by 0 or more operands |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q29:

|  |
| --- |
| The JVM has never been implemented in hardware |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q30:

|  |
| --- |
| No hardware JVM has been a commercial success |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q31:

|  |
| --- |
| The runtime constant pool for a class contains the bytecode for each of its methods |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q32:

|  |
| --- |
| A JVM address is 32 bits allowing up to 4 gigabytes of memory |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q33:

|  |
| --- |
| Metadata about classes is, since Java 8, stored in the metaspace area rather than the heap |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q34:

|  |
| --- |
| Before or during invocation, a method’s symbolic reference is resolved and replaced with a pointer |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q35:

|  |
| --- |
| The *finalise* method is called when the garbage collector has decided the object can be de-allocated |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q36:

|  |
| --- |
| The stack frame has a reference to the runtime constant pool of the current method’s class |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q37:

|  |
| --- |
| Java is both a language and a platform |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q38:

|  |
| --- |
| When a long value is pushed it occupies two words on the operand stack  Long’s values are 64-bit signed integers. So yeah, it occupies two words on the operand stack.  <https://docs.oracle.com/javase/specs/jvms/se8/html/jvms-2.html> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q39:

|  |
| --- |
| The JVM was not designed to be implemented in hardware |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q40:

|  |
| --- |
| Native methods, where supported, also use the current thread’s JVM stack |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q41:

|  |
| --- |
| The *invokevirtual*instruction is used for instance methods  <https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html#jvms-6.5.invokevirtual> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q42:

|  |
| --- |
| The JVM's stack provides working storage for calculations |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q43:

|  |
| --- |
| Each stack frame has a reference to the constant pool |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q44:

|  |
| --- |
| The arguments of a class method are local variables 1, 2 and so on  See Q7. Really don’t get it. |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q45:

|  |
| --- |
| Each method’s local variables are stored on the stack |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q46:

|  |
| --- |
| The Java instruction *goto* is followed by a 2 byte signed offset  <https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html#jvms-6.5.goto> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q47:

|  |
| --- |
| The JVM has one stack for each running thread |
| |  |  | | --- | --- | | Correct Answer: | Correct False | |

Q48:

|  |
| --- |
| The arguments of an instance method are local variables 1, 2 and so on  Q44 & Q7? |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q49:

|  |
| --- |
| The heap is used for allocating class instances and arrays at run-time |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q50:

|  |
| --- |
| Just-in-time compilers translate Java to JVM byte code |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |

Q51:

|  |
| --- |
| The *invokestatic*instruction is used for class methods  <https://docs.oracle.com/javase/specs/jvms/se7/html/jvms-6.html#jvms-6.5.invokestatic> |
| |  |  | | --- | --- | | Correct Answer: | Correct True | |