OCP - Test

1. What is the result of the following code?

```
1:    public class Employee {
2:       public int employeeId;
3:       public String firstName, lastName;
4:       public int yearStarted;
5:       @Override public int hashCode() {
6:          return employeeId;
7:       }
8:       public boolean equals(Employee e) {
9:          return this.employeeId == e.employeeId;
10:      }
11:      public static void main(String[] args) {
12:         Employee one = new Employee();
13:         one.employeeId = 101;
14:         Employee two = new Employee();
15:         two.employeeId = 101;
16:         if (one.equals(two)) System.out.println("Success");
17:         else System.out.println("Failure");
18:     } }
```

A. Success
B. Failure
C. The hashCode() method fails to compile.
D. The equals() method fails to compile.
E. Another line of code fails to compile.
F. A runtime exception is thrown.

Z komentarzem [1]: A. Based on the equals() method in the code, objects are equal if they have the same employeeId. The hashCode() method correctly overrides the one from Object. The equals() method is an overload of the one from Object and not an override. It would be better to pass Object since an override would be better to use here. It is odd to override hashCode() and not equals().

2. What is the output of the following?

```
Stream<String> stream = Stream.iterate("", (s) -> s + "1");
System.out.println(stream.limit(2).map(x -> x + "2"));
```

A. 12112
B. 212
C. 212112
D. java.util.stream.ReferencePipeline$3@4517d9a3
E. The code does not compile.
F. An exception is thrown.
G. The code hangs.

Z komentarzem [2]: D. No terminal operation is called, so the stream never executes. The methods chain to create a stream that would contain “2” and “12.” The first line creates an infinite stream. The second line would get the first two elements from that infinite stream and map each element to add an extra character.

3. Which of the following creates valid locales, assuming that the language and country codes follow standard conventions? (Choose all that apply.)

A. new Locale("hi");
B. new Locale("hi", "IN");
C. new Locale("IN");
D. new Locale("IN", "hi");
E. Locale.create("hi");
F. Locale.create("IN");

Z komentarzem [3]: A, B. Choices E and F are incorrect because a Locale is created using a constructor. The convention is to use lowercase for a language code and uppercase for a country code. The language is mandatory when using a constructor, which makes choices A and B correct.

4. Which of the following pairs fills in the blanks to make this code compile?

```
5:    public void read() ___________ SQLException {
6:       ___________ new SQLException();
7:   }
```

A. throw on line 5 and throw on line 6
B. throw on line 5 and throws on line 6
C. throws on line 5 and throw on line 6
D. throws on line 5 and throws on line 6
E. None of the above. SQLException is a checked exception and cannot be thrown.
F. None of the above. SQLException is a runtime exception and cannot be thrown.

Z komentarzem [4]: C. The method should declare that it throws an exception and the body of the method actually would throw it. Options E and F are incorrect because both checked and unchecked (runtime) exceptions can be declared in a method signature. Also, option F is incorrect because SQLException is a checked exception.

5. Given an instance of a Stream, s, and a Collection, c, which are valid ways of creating a parallel stream? (Choose all that apply.)

A. new ParallelStream(s)
B. c.parallel()
C. s.parallelStream()
D. c.parallelStream()
E. new ParallelStream(c)
F. s.parallel()

Z komentarzem [5]: D, F. There is no such class as ParallelStream, so A and E are incorrect. The method defined in the Stream class to create a parallel stream from an existing stream is parallel(); therefore F is correct and C is incorrect. The method defined in the Collection class to create a parallel stream from a collection is parallelStream(); therefore D is correct and B is incorrect.

6. Which classes will allow the following to compile? (Choose all that apply.)

```
InputStream is = new BufferedInputStream(new
     FileInputStream("zoo.txt"));
InputStream wrapper = new _________________ (is);
```

A. BufferedInputStream
B. FileInputStream
C. BufferedWriter
D. ObjectInputStream
E. ObjectOutputStream
F. BufferedReader

Z komentarzem [6]: A, D. The reference is for an InputStream object, so only a high-level input Stream class is permitted. B is incorrect because FileInputStream is a low-level stream that interacts directly with a file resource, not a stream resource. C and F are incorrect because you cannot use BufferedReader/BufferedWriter directly on a stream. E is incorrect because the reference is to an InputStream, not an OutputStream. A and D are the only correct options. Note that a BufferedInputStream can be wrapped twice, since high-level streams can take other high-level streams.

7. What is the output of the following code?

Z komentarzem [7]: F. The code snippet will not compile due to a bug on the first and second lines. The first line should use Paths.get(), because there is no method Path.get(). The second line passes a String to relativize() instead of a Path object. If both lines were corrected to use Paths.get(), then the correct answer would be A. Remember that the normalize() method, like most methods in the Path interface, does not modify the Path object, but instead it returns a new Path object. If it was corrected to reassign the new value to the existing path variable, then E would be correct.

```
Path path = Path.get("/user/.././root","../kodiacbear.txt");
path.normalize().relativize("/lion");
System.out.println(path);
```

A. /user/.././root/../kodiacbear.txt
B. /user/./root/kodiacbear.txt/lion
C. /kodiacbear.txt
D. kodiacbear.txt
E. ../lion
F. The code does not compile.

8. Which interfaces or classes are in a database-specific JAR file? (Choose all that apply.)

A. Driver
B. Driver's implementation
C. DriverManager
D. DriverManager's implementation
E. Statement
F. Statement's implementation

Z komentarzem [8]: B, F. The Driver, Connection, Statement, and ResultSet interfaces are part of the JDK, making choices A and E incorrect. The concrete DriverManager class is also part of the JDK, making choices C and D incorrect. Choices B and F are correct since the implementation of these interfaces is part of the database-specific driver JAR file.

9. **What is the result of compiling the following class?**

```
public class Book {
   private int ISBN;
   private String title, author;
   private int pageCount;
   public int hashCode() {
      return ISBN;
   }
   @Override public boolean equals(Object obj) {
     if ( !(obj instanceof Book)) {
        return false;
     }
     Book other = (Book) obj;
     return this.ISBN == other.ISBN;
   }
// imagine getters and setters are here
}
```

A. The code compiles.
B. The code does not compile because hashCode() is incorrect.
C. The code does not compile because equals() does not override the parent method correctly.
D. The code does not compile because equals() tries to refer to a private field.
E. The code does not compile because the ClassCastException is not handled or declared.

Z komentarzem [9]: A. hashCode() is correct and perfectly reasonable given that equals() also checks that field. ClassCastException is a runtime exception and therefore does not need to be handled or declared. The override in equals() is correct. It is common for equals() to refer to a private instance variable. This is legal because it is within the same class, even if it is referring to a different object of the same class.

F. The code does not compile for another reason.

10. Suppose that you need to work with a collection of elements that need to be sorted in their natural order, and each element has a unique string associated with its value. Which of the following collections classes in the java.util package best suit your needs for this scenario?

A. ArrayList
B. HashMap
C. HashSet
D. TreeMap
E. TreeSet
F. Vector

Z komentarzem [10]: E. The code does not compile because EasternChipmunk inherits the abstract method climb() but does not implement it, therefore the correct answer is E. B, C, and D are incorrect as they compile for various reasons. Line 2 compiles, as non-static and non-default interface methods are assumed to have the abstract modifier. Line 4 compiles without issue as an interface can extend another interface. Line 5 compiles without issue as an abstract class can implement an interface without implementing any of the abstract methods. F is incorrect, as Line 8 does not compile.

11. What is the result of the following code?

```
1: public interface CanClimb {
2:    public abstract void climb();
3: }
4: public interface CanClimbTrees extends CanClimb {}
5: public abstract class Chipmunk implements CanClimbTrees {
6:    public abstract void chew();
7: }
8: public class EasternChipmunk extends Chipmunk {
9:    public void chew() { System.out.println("Eastern Chipmunk is
Chewing"); }
10: }
```

A. It compiles and runs without issue.
B. The code will not compile because of line 2.
C. The code will not compile because of line 4.
D. The code will not compile because of line 5.
E. The code will not compile because of line 8.
F. It compiles but throws an exception at runtime.

12. Which of the following changes when made independently would make this code compile? (Choose all that apply.)

```
1:    public class StuckTurkeyCage implements AutoCloseable {
2:       public void close() throws Exception {
3:          throw new Exception("Cage door does not close");
4:       }
5:       public static void main(String[] args) {
6:          try (StuckTurkeyCage t = new StuckTurkeyCage()) {
7:            System.out.println("put turkeys in");
8:          }
9:       }
10:   }
```

A. Remove throws Exception from the declaration on line 2.
B. Add throws Exception to the declaration on line 5.
C. Change line 8 to } catch (Exception e) {}.
D. Change line 8 to } finally {}.
E. None of the above will make the code compile.
F. The code already compiles as is.

Z komentarzem [11]: D. The answer needs to implement Map because you are dealing with key/value pairs per the unique string text. You can eliminate A, C, E, and F immediately. ArrayList and Vector are Lists. HashSet and TreeSet are Sets. Now it is between HashMap and TreeMap. Since the question talks about ordering, you need the TreeMap. Therefore, the answer is E.

13. Which are required parts of a JDBC URL? (Choose all that apply.)

A. Connection parameters
B. Database name
C. jdbc
D. Location of database
E. Port
F. Password

14. What is the result of the following code?

```
String s1 = "Canada";
String s2 = new String(s1);
if(s1 == s2) System.out.println("s1 == s2");
if(s1.equals(s2)) System.out.println("s1.equals(s2)");
```

A. There is no output.
B. s1 == s2
C. s1.equals(s2)
D. Both B and C.
E. The code does not compile.
F. The code throws a runtime exception.

Z komentarzem [12]: C. s1 points to the string pool. s2 points to an object on the heap, since it is created at runtime. == checks for reference equality. These are different references, making B incorrect. String overrides equals() so the actual values are the same, making C correct. And yes, this question could have appeared on the OCA. Remember that the OCP is cumulative. A question may appear to be about one thing and actually be about a simpler concept.

15. What is the result of the following statements?

```
3:    List list = new ArrayList();
4:    list.add("one");
5:    list.add("two");
6:    list.add(7);
7:    for (String s: list)
8:    System.out.print(s);
```

A. onetwo
B. onetwo7
C. onetwo followed by an exception
D. Compiler error on line 6
E. Compiler error on line 7

Z komentarzem [13]: A, D. A is correct as Climb defines an interface with exactly one abstract method. B is incorrect, as abstract classes are not functional interfaces despite having a single abstract method. While functional interfaces may have any number of default methods, ArcticMountainClimb will not compile due to the default method getSpeed() missing an implementation body, so C is incorrect. D is correct, as the interface MountainClimb has exactly one abstract method defined in Climb. Finally, E is incorrect because A and D are correct.

16. Which of the following are valid functional interfaces? (Choose all that apply.)

```
public interface Climb {
   public int climb();
}
public abstract class Swim {
   public abstract Object swim(double speed, int duration);
}
public interface ArcticMountainClimb extends MountainClimb {
   public default int getSpeed();
}
public interface MountainClimb extends Climb {}
```

A. Climb
B. Swim
C. ArcticMountainClimb
D. MountainClimb
E. None of these are valid functional interfaces.

Z komentarzem [14]: E. The code does not compile. It attempts to mix generics and legacy code. Lines 3 through 7 create an ArrayList without generics. This means that we can put any objects in it. Line 7 should be looping through a list of Objects rather than Strings since we didn't use generics.

17. Which of the following are true? (Choose all that apply.)

A. All keys must be in the same resource bundle file to be used.
B. All resource bundles defined as Java classes can be expressed using the property file format instead.
C. All resource bundles defined as property files can be expressed using the Java class list bundle format instead.
D. Changing the default locale lasts for only a single run of the program.
E. It is forbidden to have both Props_en.java and Props_en.properties in the classpath of an application.

Z komentarzem [15]: C, D. Choice A is incorrect because Java will look at parent bundles. For example, Java will look at Props.properties if Props_en.properties does not contain the requested key. Java class resource bundles can have non-String values while property files are limited to strings. Therefore, choice B is incorrect and choice C is correct. Choice D is correct because the locale is only changed in memory. Choice E is incorrect because Java specifies that it will look for a Java class resource bundle before a property file of the same name.

18. What is the output of the following?

```
Predicate<? super String> predicate = s -> s.length() > 3;
Stream<String> stream = Stream.iterate("-", (s) -> s + s);
boolean b1 = stream.noneMatch(predicate);
boolean b2 = stream.anyMatch(predicate);
System.out.println(b1 + " " + b2);
```

A. false true
B. false false
C. java.util.stream.ReferencePipeline$3@4517d9a3
D. The code does not compile.
E. An exception is thrown.
F. The code hangs.

Z komentarzem [16]: E. An infinite stream is generated where each element is twice as long as the previous one. b1 is set to false because Java finds an element that doesn't match when it gets to the element of length 4. However, the next line tries to operate on the same stream. Since streams can be used only once, this throws an exception that the "stream has already been operated upon or closed." If two different streams were used, the result would be option A.

19. Which of the following fills in the blank to make the code compile? (Choose all that apply)

```
public static void main(String[] args)  {
   try {
      throw new IOException();
   } catch (_____________________________ ) { }
}
```

A. FileNotFoundException | IOException e
B. FileNotFoundException e | IOException e
C. FileNotFoundException | RuntimeException e
D. FileNotFoundException e | RuntimeException e
E. IOException | RuntimeException e
F. IOException e | RuntimeException e

Z komentarzem [17]: E. Options B, D, and F are incorrect because only one variable name is allowed in a multi-catch block. Option A is incorrect because FileNotFoundException is a subclass of IOException. A multi-catch statement does not allow redundancy, and just catching IOException would have been equivalent. Option C is incorrect because the IOException that is thrown is not handled.

20. What is the result of executing the following code? (Choose all that apply.)

```
1: Path path = Paths.get("sloth.schedule");
2: BasicFileAttributes attributes = Files.readAttributes(path,
BasicFileAttributes.class);
3: if(attributes.size()>0 && attributes.creationTime().toMillis()>0) {
4:    attributes.setTimes(null,null,null);
5: }
```

A. It compiles and runs without issue.
B. The code will not compile because of line 2.
C. The code will not compile because of line 3.
D. The code will not compile because of line 4.
E. The code compiles but throws an exception at runtime.

Z komentarzem [18]: D. The setTimes() method is available only on BasicFileAttributeView, not the read-only BasicFileAttributes class, so line 4 will not compile and D is correct. You need to retrieve an instance of the view class to update the data. The rest of the lines compile without issue and only D is correct.

21. Which lines need to be changed to make the code compile? (Choose all that apply.)

```
ExecutorService service =
      Executors.newSingleThreadScheduledExecutor();
service.scheduleWithFixedDelay(() -> { // w1
       System.out.println("Open Zoo");
       return null; // w2
   }, 0, 1, TimeUnit.MINUTES);
Future<?> result =
      service.submit(() -> System.out.println("Wake Staff")); // w3
System.out.println(result.get()); // w4
```

A. It compiles and runs without issue.
B. Line w1
C. Line w2
D. Line w3
E. Line w4
F. It compiles but throws an exception at runtime.

Z komentarzem [19]: B, C. The code does not compile, so A and F are incorrect. The first problem is that although a ScheduledExecutorService is created, it is assigned to an ExecutorService. Since scheduleWithFixedDelay() does not exist in ExecutorService, line w1 will not compile, and B is correct. The second problem is that scheduleWithFixedDelay()supports only Runnable, not Callable, and any attempt to return a value is invalid in a Runnable lambda expression; therefore line w2 will also not compile and C is correct. The rest of the lines compile without issue, so D and E are incorrect.

22. Which of the following is a valid JDBC URL?

A. jdbc:sybase:localhost:1234/db
B. jdbc::sybase::localhost::/db
C. jdbc::sybase:localhost::1234/db
D. sybase:localhost:1234/db
E. sybase::localhost::/db
F. sybase::localhost::1234/db

Z komentarzem [20]: A. A JDBC URL has three main parts separated by single colons, making choices B, C, E, and F incorrect. The first part is always jdbc. Therefore, the correct answer is A. Notice that you can get this right even if you've never heard of the Sybase database before.

23. What is true about the following code? You may assume city and mascot are never null.

```
public class BaseballTeam {
   private String city, mascot;
   private int numberOfPlayers;
   public boolean equals(Object obj) {
      if ( !(obj instanceof BaseballTeam))
        return false;
      BaseballTeam other = (BaseballTeam) obj;
      return (city.equals(other.city) &&
mascot.equals(other.mascot));
   }
   public int hashCode() {
      return numberOfPlayers;
   }
// imagine getters and setters are here
}
```

A. The class does not compile.
B. The class compiles but has an improper equals() method.
C. The class compiles but has an improper hashCode() method.
D. The class compiles and has proper equals() and hashCode() methods.

Z komentarzem [21]: C. The equals() method is correct. You are allowed to use any business logic that you want in determining equality. The hashCode() method is not correct. It violates the rule that two objects that return true for equals() must return the same hashCode(). It is also a bad idea for the hash code to contain values that could change.

24. Which of the following are valid lambda expressions? (Choose all that apply.)

A. () -> ""
B. x,y -> x+y
C. (Coyote y) -> return 0;
D. (Camel c) -> {return;}
E. Wolf w -> 39
F. () ->
G. (Animal z, m) -> a

Z komentarzem [22]: E. Since we call push() rather than offer(), we are treating the ArrayDeque as a LIFO (last-in, first-out) stack. On line 7, we remove the last element added, which is "ola". On line 8, we look at the new last element ("hi"), but don't remove it. Lines 9 and 10, we remove each element in turn until none are left. Note that we don't use an Iterator to loop through the ArrayDeque. The order in which the elements are stored internally is not part of the API contract.

25. Which are true statements about terminal operations in a stream? (Choose all that apply.)

A. At most one terminal operation can exist in a stream pipeline.
B. Terminal operations are a required part of the stream pipeline in order to get a result.
C. Terminal operations have Stream as the return type.
D. The referenced Stream may be used after the calling a terminal operation.
E. The peek() method is an example of a terminal operation.

Z komentarzem [23]: A, B. Terminal operations are the final step in a stream pipeline. Exactly one is required, because it triggers the execution of the entire stream pipeline. Therefore, options A and B are correct. Options C and F are true of intermediate operations rather than terminal operations. Option E is never true. Once a stream pipeline is run, the Stream is marked invalid.

26. Assume that all bundles mentioned in the answers exist and define the same keys. Which one will be used to find the key in line 8?

```
6:    Locale.setDefault(new Locale("en", "US"));
7:    ResourceBundle b = ResourceBundle.getBundle("Dolphins");
8:    b.getString("name");
```

A. Dolphins.properties
B. Dolphins_en.java
C. Dolphins_en.properties
D. Whales.properties
E. Whales_en_US.properties
F. The code does not compile.

Z komentarzem [24]: B. Java will first look for the most specific matches it can find, starting with Dolphins_en_US.java and then Dolphins_en_US.properties. Since neither is found, it drops the country and looks for Dolphins_en.java. Since a match is found, there is no reason to go on to the next one, which is Dolphins_en.properties.

27. Which of the following are true statements? (Choose all that apply.)

A. A traditional try statement without a catch block requires a finally block.
B. A traditional try statement without a finally block requires a catch block.
C. A traditional try statement with only one statement can omit the {}.
D. A try-with-resources statement without a catch block requires a finally block.
E. A try-with-resources statement without a finally block requires a catch block.
F. A try-with-resources statement with only one statement can omit the {}.

Z komentarzem [25]: A, B. A try-with-resources statement does not require a catch or finally block. A traditional try statement requires at least one of the two.

28. What statement about the following code is true?

```
AtomicLong value1 = new AtomicLong(0);
final long[] value2 = {0};
IntStream.iterate(1, i -> 1).limit(100).parallel()
    .forEach(i -> value1.incrementAndGet());
IntStream.iterate(1, i -> 1).limit(100).parallel()
         .forEach(i -> ++value2[0]);
System.out.println(value1+" "+value2[0]);
```

A. It outputs 100 100.
B. It outputs 100 99.
C. The output cannot be determined ahead of time.
D. The code does not compile.
E. It compiles but throws an exception at runtime.
F. It compiles but enters an infinite loop at runtime.

Z komentarzem [26]: C. The code compiles and runs without throwing an exception or entering an infinite loop, so D, E, and F are incorrect. The key here is that the increment operator ++ is not atomic. While the first part of the output will always be 100, the second part is nondeterministic. It could output any value from 1 to 100, because the threads can overwrite each other's work. Therefore, C is the correct answer and A and B are incorrect.

29. Which of the following can fill in the blank to make the code compile? (Choose all that apply.)

```
Console c = System.console();
String s = _____;
```

A. c.input()
B. c.read()
C. c.readLine()
D. c.readPassword()
E. c.readString()
F. None of the above

Z komentarzem [27]: C. The readLine() method returns a String and reads a line of input from the console. readPassword() returns a char[]. The others do not exist.

30. If the current working directory is /user/home, then what is the output of the following code?

```
Path path = Paths.get("/zoo/animals/bear/koala/food.txt");
System.out.println(path.subpath(1,3).getName(1).toAbsolutePath());
```

A. animals/bear
B. koala
C. /user/home/bear
D. /user/home/koala/koala
E. /user/home/food.txt
F. /user/home/koala/food.txt
G. The code does not compile.

Z komentarzem [28]: C. First off, the code compiles without issue, so G is incorrect. Let's take this one step at a time. First, the subpath() method is applied to the absolute path, which returns the relative path animals/bear. Next, the getName() method is applied to the relative path, and since this is indexed from zero, it returns the relative path bear. Finally, the toAbsolutePath() method is applied to the relative path bear, resulting in the current directory being incorporated into the path. The final output is the absolute path /user/home/bear, so C is correct.

31. What file is required inside a JDBC 4.0+ driver JAR?

A. java.sql.Driver
B. META-INF/java.sql.Driver
C. META-INF/db/java.sql.Driver
D. META-INF/database/java.sql.Driver
E. META-INF/service/java.sql.Driver

Z komentarzem [29]: E. Starting with JDBC 4.0, driver implementations were required to provide the name of the class implementing Driver in a file named java.sql.Driver in the directory META-INF/service.

32. Which of the following statements are true, assuming a and b are String objects? (Choose all that apply.)

A. If a.equals(b) is true, a.hashCode() == b.hashCode() is always true.
B. If a.equals(b) is true, a.hashCode() == b.hashCode() is sometimes but not always true.
C. If a.equals(b) is false, a.hashCode() == b.hashCode() can never be true.
D. If a.equals(b) is false, a.hashCode() == b.hashCode() can sometimes be true.

Z komentarzem [30]: A, D. The relevant rule is that two objects that return true for equals() objects must return the same hash code. Therefore A is correct and B is incorrect. Two objects with the same hash code may or may not be equal. This makes C incorrect and D correct. The fact that two objects are not equal does not guarantee or preclude them from sharing a hash code. Remember that hashCode() tells you which bucket to look in and equals() tells you whether you have found an exact match.

33. Suppose that we have the following property files and code. Which bundles are used on lines 8 and 9 respectively?

```
Dolphins.properties
name=The Dolphin
age=0

Dolphins_en.properties
name=Dolly
age=4

Dolphins_fr.properties
name=Dolly

5:    Locale fr = new Locale("fr");
6:    Locale.setDefault(new Locale("en", "US"));
7:    ResourceBundle b = ResourceBundle.getBundle("Dolphins", fr);
8:    b.getString("name");
9:    b.getString("age");
```

A. Dolphins.properties and Dolphins.properties
B. Dolphins.properties and Dolphins_en.properties
C. Dolphins_en.properties and Dolphins_en.properties
D. Dolphins_fr.properties and Dolphins.properties
E. Dolphins_fr.properties and Dolphins_en.properties
F. The code does not compile.

Z komentarzem [31]: D. Java will use Dolphins_fr.properties as the matching resource bundle on line 7 because it is an exact match on the language of the requested locale. Line 8 finds a matching key in this file. Line 9 does not find a match in that file, and therefore it has to look higher up in the hierarchy. Once a bundle is chosen, only resources in that hierarchy are allowed.

34. What is the output of the following code?

```
import java.io.*;
public class AutocloseableFlow {
   static class Door implements AutoCloseable {
      public void close() {
         System.out.print("D");
      } }
   static class Window implements Closeable {
      public void close() {
         System.out.print("W");
         throw new RuntimeException();
   } }
   public static void main(String[] args) {
      try (Door d = new Door(); Window w = new Window()) {
         System.out.print("T");
      } catch (Exception e) {
         System.out.print("E");
      } finally {
         System.out.print("F");
      } } }
```

A. TWF
B. TWDF
C. TWDEF
D. TWF followed by an exception
E. TWDF followed by an exception
F. TWEF followed by an exception
G. The code does not compile.

Z komentarzem [32]: C. After opening both resources in the try-with-resources statement, T is printed. Then the try-with-resource completes and closes the resources in reverse order from which they were opened. After W is printed, an exception is thrown. However, the remaining resource is still closed and D is printed. The exception thrown is then caught and E is printed. Last, the finally block is run, printing F. Therefore the answer is TWDEF.

35. What is the result of executing the following code? (Choose all that apply.)

```
String line;
Console c = System.console();
Writer w = c.writer();
if ((line = c.readLine()) != null)
   w.append(line);
w.flush();
```

A. The code runs without error but prints nothing.
B. The code prints what was entered by the user.
C. An ArrayIndexOutOfBoundsException might be thrown.
D. A NullPointerException might be thrown.
E. An IOException might be thrown.
F. The code does not compile.

Z komentarzem [33]: B, D, E. This is correct code for reading a line from the console and writing it back out to the console, making option B correct. Options D and E are also correct. If no console is available, a NullPointerException is thrown. The append() method throws an IOException.

36. Assume /kang exists as a symbolic link to the directory /mammal/kangaroo within the file system. Which of the following statements are correct about this code snippet? (Choose all that apply.)

```
Path path = Paths.get("/kang");
if(Files.isDirectory(path) && Files.isSymbolicLink(path))
   Files.createDirectory(path.resolve("joey"));
```

A. A new directory will always be created.
B. A new directory will be created only if /mammal/kangaroo exists.
C. If the code creates a directory, it will be reachable at /kang/joey.
D. If the code creates a directory, it will be reachable at /mammal/kangaroo/joey.
E. The code does not compile.
F. The code will compile but always throws an exception at runtime.

Z komentarzem [34]: B, C, D. The first clause of the if/then statement will be true only if the target of the symbolic link, /mammal/kangaroo, exists, since by default isDirectory() follows symbolic links, so B is correct. Option A is incorrect because /mammal/kangaroo may not exist or /mammal/kangaroo/joey may already exist. If /mammal/kangaroo does exist, then the directory will be created at /mammal/kangaroo/joey, and because the symbolic link would be accessible as /kang/joey, C and D are both correct. E is incorrect, because the code compiles without issue. F is incorrect because the code may throw an exception at runtime, such as when the file system is unavailable or locked for usage; thus it is not guaranteed to throw an exception at runtime.

37. Suppose that you have a table named animal with two rows. What is the result of the following code?

```
6:  Connection conn = new Connection(url, userName, password);
7:  Statement stmt = conn.createStatement();
8:  ResultSet rs = stmt.executeQuery("select count(*) from animal");
9:  if (rs.next()) System.out.println(rs.getInt(1));
```

A. 0
B. 2
C. There is a compiler error on line 6.
D. There is a compiler error on line 9.
E. There is a compiler error on another line.
F. A runtime exception is thrown.

Z komentarzem [35]: C. A Connection is created using a static method on DriverManager. It does not use a constructor. Therefore, choice C is correct. If the Connection was created properly, the answer would be choice B.

38. What is the result of the following code?

```
public class FlavorsEnum {
   enum Flavors { VANILLA, CHOCOLATE, STRAWBERRY }
   public static void main(String[] args) {
      System.out.println(Flavors.CHOCOLATE.ordinal());
   }
}
```

A. 0
B. 1
C. 9
D. CHOCOLATE
E. The code does not compile due to a missing semicolon.
F. The code does not compile for a different reason.

Z komentarzem [36]: B. The ordinal() method of an enum returns its corresponding int value. Like arrays, enums are zero based. Remember that the index of an enum may change when you recompile the code and should not be used for comparison.

39. What is the result of the following code?

Z komentarzem [37]: A. This code compiles and runs without issue so C, D, E, and F are incorrect. Line h1 creates a lambda expression that checks if the age is less than 5. Since there is only one parameter and it does not specify a type, the parentheses around the type parameter are optional. Line h2 uses the Predicate interface, which declares a test() method. Since test() returns true on the expression, match is output and A is correct.

```
1:    public class Hello<T> {
2:       T t;
3:       public Hello(T t) { this.t = t; }
4:       public String toString() { return t.toString(); }
5:       public static void main(String[] args) {
6:          System.out.print(new Hello<String>("hi"));
7:          System.out.print(new Hello("there"));
8:       } }
```

A. hi
B. hi followed by a runtime exception
C. hithere
D. Compiler error on line 4
E. Compiler error on line 6
F. Compiler error on line 7

40. What is the result of the following class?

Z komentarzem [38]: C. Line 7 gives a compiler warning for not using generics but not a compiler error. Line 4 compiles fine because toString() is defined on the Object class and is therefore always available to call. Line 6 creates the Hello class with the generic type String. Line 7 creates the Hello class with the generic type Object since no type is specified.

```
import java.util.function.*;
public class Panda {
   int age;
   public static void main(String[] args) {
      Panda p1 = new Panda();
      p1.age = 1;
      check(p1, p -> p.age < 5);  // h1
   }
   private static void check(Panda panda, Predicate<Panda> pred) {
// h2
      String result = pred.test(panda) ? "match": "not match";  //
h3
      System.out.print(result);
} }
```

A. match
B. not match
C. Compiler error on line h1.
D. Compiler error on line h2.
E. Compile error on line h3.
F. A runtime exception is thrown.

41. Which of the following can fill in the blank so that the code prints out false? (Choose all that apply.)

```
Stream<String> s = Stream.generate(() -> "meow");
boolean match = s._________________(String::isEmpty);
System.out.println(match);
```

A. allMatch
B. anyMatch
C. findAny
D. findFirst
E. noneMatch
F. None of the above

42. Which of the following can be inserted into the blank to create a date of June 21, 2014? (Choose all that apply.)

```
import java.time.*;

public class StartOfSummer {
  public static void main(String[] args) {
  LocalDate date = ____________________________
} }
```

A. new LocalDate(2014, 5, 21);
B. new LocalDate(2014, 6, 21);
C. LocalDate.of(2014, 5, 21);
D. LocalDate.of(2014, 6, 21);
E. LocalDate.of(2014, Calendar.JUNE, 21);
F. LocalDate.of(2014, Month.JUNE, 21);

43. Which happens when more tasks are submitted to a thread executor than available threads?

A. The thread executor will throw an exception when a task is submitted that is over its thread limit.
B. The task will be added to an internal queue and completed when there is an available thread.
C. The thread executor will discard any task over its thread limit.
D. The call to submit the task to the thread executor will wait until there is a thread available before continuing.
E. The thread executor creates new temporary threads to complete the additional tasks.

Z komentarzem [39]: A. Options C and D are incorrect because these methods do not take a Predicate parameter and do not return a boolean. Options B and E are incorrect because they cause the code to run infinitely. The stream has no way to know that a match won't show up later. Option A is correct because it is safe to return false as soon as one element passes through the stream that doesn't match.

Z komentarzem [40]: D, F. Options A and B are incorrect because LocalDate does not have a public constructor. Option C is incorrect because months start counting with 1 rather than 0. Option E is incorrect because it uses the old Calendar constants for months, which begin with 0. Options D and F are both correct ways of specifying the desired date.

Z komentarzem [41]: B. If a task is submitted to a thread executor, and the thread executor does not have any available threads, the call to the task will return immediately with the task being queued internally by the thread executor. For this reason, B is the only correct answer.

44. What is the output of the following code?

```
import java.io.*;
public class AutocloseableFlow {
   static class Door implements AutoCloseable {
      public void close() {
         System.out.print("D");
         throw new RuntimeException();
      } }
   static class Window implements Closeable {
      public void close() {
         System.out.print("W");
         throw new RuntimeException();
   } }
   public static void main(String[] args) {
      try {
        Door d = new Door(); Window w = new Window()
      }
      {
         System.out.print("T");
      } catch (Exception e) {
         System.out.print("E");
      } finally {
         System.out.print("F");
      } } }
```

A. TWF
B. TWDF
C. TWDEF
D. TWF followed by an exception
E. TWDF followed by an exception
F. TWEF followed by an exception
G. The code does not compile.

Z komentarzem [42]: G. A try-with-resources statement uses parentheses rather than brackets for the try section. This is likely subtler than a question that you'll get on the exam, but it is still important to be on alert for details.

45. Which of the following are true statements about serialization in Java? (Choose all that apply.)

A. The process of converting serialized data back into memory is called deserialization.
B. All non-thread classes should be marked Serializable.
C. The Serializable interface requires implementing serialize() and deserialize() methods.
D. The Serializable interface is marked final and cannot be extended.
E. The readObject() method of ObjectInputStream may throw a ClassNotFoundException even if the return object is not explicitly cast.

Z komentarzem [43]: A, E. The first statement is the definition of deserialization, so A is correct. B is incorrect, because you may mark (or not mark) a class as serializable for a variety of reasons. C is incorrect because the Serializable interface has no method requirements, and any class can implement the interface. D is also incorrect, because the Serializable interface may be extended by your own interface. Finally, E is correct, because the exception may be thrown within the readObject() even if the result is not cast.

46. Given that /animals is a directory that exists and it is empty, what is the result of the following code?

Z komentarzem [44]: C. The code does not compile since the stream output by Files.walk() is Stream<Path>, therefore we need a Predictate, not a BiPredicate, on line w1, and the answer is C. If the Files.find() method had been used instead, and the lambda had been passed as an argument to the method instead of on filter(), the output would be B, Has Sub-directory, since the directory is given to be empty. For fun, we reversed the expected output of the ternary operation to make sure that you understood the process.

```
Path path = Paths.get("/animals");
boolean myBoolean = Files.walk(path)
   .filter((p,a) -> a.isDirectory() && !path.equals(p)) // w1
   .findFirst().isPresent();  // w2
System.out.println(myBoolean ? "No Sub-directory": "Has Sub-
directory");
```

A. It prints No Sub-directory.
B. It prints Has Sub-directory.
C. The code will not compile because of line w1.
D. The code will not compile because of line w2.
E. The output cannot be determined.
F. It produces an infinite loop at runtime.

47. What is the result of the following code? (Choose all that apply.)

```
public class IceCream {
   enum Flavors { VANILLA, CHOCOLATE, STRAWBERRY }
   public static void main(String[] args) {
      Flavors f = Flavors.STRAWBERRY;
      switch (f) {
         case 0: System.out.println("vanilla");
         case 1: System.out.println("chocolate");
         case 2: System.out.println("strawberry");
              break;
         default: System.out.println("missing flavor");
     } } }
```

A. vanilla
B. chocolate
C. strawberry
D. missing flavor
E. The code does not compile.
F. An exception is thrown.

48. Which of the following statements are true? (Select two.)

```
3:    Set<Number> numbers = new HashSet<>();
4:    numbers.add(new Integer(86));
5:    numbers.add(75);
6:    numbers.add(new Integer(86));
7:    numbers.add(null);
8:    numbers.add(309L);
9:    Iterator iter = numbers.iterator();
10:   while (iter.hasNext())
11:      System.out.print(iter.next());
```

Z komentarzem [45]: E. A case statement on an enum data type must be the unqualified name of an enumeration constant. For example, case VANILLA would be valid. You cannot use the ordinal equivalents. Therefore, the code does not compile.

Z komentarzem [46]: C, E, G, H. A is incorrect, as there are definitely some problems with the immutable objects implementation. B is incorrect, as there is no such thing as the Immutable interface defined in the Java API. C is correct, as all instance variables should be private and final to prevent them from being changed by a caller. D is incorrect, as adding settings is the opposite of what you do with the immutable object pattern. E is correct, since List<Seal> is mutable, all direct access should be removed. F is incorrect, as this has nothing to do with immutability. G is correct, as we need to copy the mutable List<Seal> to prevent the caller of the constructor from maintaining access to a mutable structure within our class. H is also correct, as it prevents the methods of the class from being overridden.

A. The code compiles successfully.
B. The output is 8675null309.
C. The output is 867586null309.
D. The output is indeterminate.
E. There is a compiler error on line 3.
F. There is a compiler error on line 9.
G. An exception is thrown.

49. What changes need to be made to make the following immutable object pattern correct? (Choose all that apply.)

```
import java.util.List;
public class Seal {
   String name;
   private final List<Seal> friends;
   public Seal(String name, List<Seal> friends) {
      this.name = name;
      this.friends = friends;
   }
   public String getName() { return name; }
   public List<Seal> getFriends() { return friends; }
}
```

A. None; the immutable object pattern is properly implemented.
B. Have Seal implement the Immutable interface.
C. Mark name final and private.
D. Add setters for name and List<Seal> friends.
E. Replace the getFriends() method with methods that do not give the caller direct access to the List<Seal> friends.
F. Change the type of List<Seal> to List<Object>.
G. Make a copy of the List<Seal> friends in the constructor.
H. Mark the Seal class final.

50. We have a method that returns a sorted list without changing the original. Which of the following can replace the method implementation to do the same with streams?

```
private static List<String> sort(List<String> list) {
   List<String> copy = new ArrayList<>(list);
   Collections.sort(copy, (a, b) -> b.compareTo(a));
   return copy;
}
```

A. return list.stream().compare((a, b) -> b.compareTo(a)).collect(Collectors.toList());
B. return list.stream().compare((a, b) -> b.compareTo(a)).sort();
C. return list.stream().compareTo((a, b) -> b.compareTo(a)).collect(Collectors.toList());
D. return list.stream().compareTo((a, b) -> b.compareTo(a)).sort();
E. return list.stream().sorted((a, b) -> b.compareTo(a)).collect();
F. return list.stream().sorted((a, b) -> b.compareTo(a)).collect(Collectors.toList());

Z komentarzem [47]: A, D. The code compiles fine. It uses the diamond operator, and it allows any implementation of Number to be added. HashSet does not guarantee any iteration order, making A and D correct.

Z komentarzem [48]: F. The sorted() method is used in a stream pipeline to return a sorted Stream. A collector is needed to turn the stream back into a List. The collect() method takes the desired collector.

51. What is the output of the following code?

```
LocalDate date = LocalDate.parse(
"2018-04-30", DateTimeFormatter.ISO_LOCAL_DATE);
date.plusDays(2);
date.plusHours(3);
System.out.println(date.getYear() + " "
  + date.getMonth() + " "+ date.getDayOfMonth());
```

A. 2018 APRIL 2
B. 2018 APRIL 30
C. 2018 MAY 2
D. The code does not compile.
E. A runtime exception is thrown.

Z komentarzem [49]: D. A LocalDate does not have a time element. Therefore, it has no method to add hours, and the code does not compile.

52. What is the result of running java EchoInput hi there with the following code?

```
public class EchoInput {
   public static void main(String [] args) {
      if(args.length <= 3) assert false;
      System.out.println(args[0] + args[1] + args[2]);
   }
}
```

A. hithere
B. The assert statement throws an AssertionError.
C. The code throws an ArrayIndexOutOfBoundsException.
D. The code compiles and runs successfully, but there is no output.
E. The code does not compile.

Z komentarzem [50]: C. The code compiles fine, so option E is incorrect. The command line has only two arguments, so args.length is 2 and the if statement is true. However, because assertions are not enabled, it does not throw an AssertionError, so option B is incorrect. The println attempts to print args[2], which generates an ArrayIndexOutOfBoundsException, so the answer is option C.

53. Fill in the blank: _____ is the topmost directory on a file system.

A. Absolute
B. Directory
C. Parent
D. Root
E. Top

Z komentarzem [51]: D. The root directory is the top-level directory; therefore D is correct. The rest of the statements are invalid or incorrect.

54. If the current working directory is /zoo, and the path /zoo/turkey does not exist, then what is the result of executing the following code? (Choose all that apply.)

```
Path path = Paths.get("turkey");
if(Files.isSameFile(path,Paths.get("/zoo/turkey")))  // x1
   Files.createDirectory(path.resolve("info"));  // x2
```

A. The code compiles and runs without issue, but it does not create any directories.

Z komentarzem [52]: F. The code compiles without issue, so D and E are incorrect. The method Files.isSameFile() first checks to see if the Path values are the same in terms of equals(). Since the first path is relative and the second path is absolute, this comparison will return false, forcing isSameFile() to check for the existence of both paths in the file system. Since we know /zoo/turkey does not exist, a NoSuchFileException is thrown and F is the correct answer. A, B, and C are incorrect since an exception is thrown at runtime.

B. The directory /zoo/turkey is created.
C. The directory /zoo/turkey/info is created.
D. The code will not compile because of line x1.
E. The code will not compile because of line x2.
F. It compiles but throws an exception at runtime.

55. What is the result of the following code?

```
TreeSet<String> tree = new TreeSet<String>();
tree.add("one");
tree.add("One");
tree.add("ONE");
System.out.println(tree.ceiling("On"));
```

A. On
B. one
C. One
D. ONE
E. The code does not compile.
F. An exception is thrown.

56. Which of the following are true of interfaces? (Choose all that apply.)

A. They can extend other classes.
B. They cannot be extended.
C. They enable classes to have multiple inheritance.
D. They can only contain abstract methods.
E. They can be declared final.
F. All members of an interface are public.

56. What is the output of the following code?

```
LocalDate date = LocalDate.of(2018, Month.APRIL, 40);
System.out.println(date.getYear() + " " + date.getMonth()
   + " "+ date.getDayOfMonth());
```

A. 2018 APRIL 4
B. 2018 APRIL 30
C. 2018 MAY 10
D. Another date
E. The code does not compile.
F. A runtime exception is thrown.

57. Which of the following command lines cause this program to fail on the assertion? (Choose all that apply.)

```
public class On {
   public static void main(String[] args) {
      String s = null;
```

Z komentarzem [53]: C, F. A and B are both incorrect as interfaces can extend other interfaces, although not classes. C is correct since a class may implement multiple interfaces. D is incorrect as interfaces have static and default methods, as well as static final variables. E is incorrect as interfaces are assumed to be abstract, and abstract and final can never be used together. F is correct as interface methods and variables are each assumed public.

Z komentarzem [54]: C. TreeSet sorts the elements. Since uppercase letters sort before lowercase letters, the ordering is "ONE", "One", "one". The ceiling() method returns the smallest element greater than the specified one. "On" appears between "ONE" and "One". Therefore, the smallest element that is larger than the specified value is "One".

Z komentarzem [55]: F. Java throws an exception if invalid date values are passed. There is no 40th day in April—or any other month for that matter.

Z komentarzem [56]: B, C. Java uses the flags –ea or –enableassertions to turn on assertions. –da or –disableassertions turns off assertions. The colon indicates for a specific class. Choice B is correct because it turns on assertions for all code. Choice C is correct because it disables assertions but then turns them back on for this class.

```
      assert s != null;
   }
}
```

A. java –da On
B. java –ea On
C. java -da -ea:On On
D. java -ea -da:On On
E. The code does not compile.

58. What statements about the following code are true? (Choose all that apply.)

```
Integer i1 = Arrays.asList(1,2,3,4,5).stream().findAny().get();
synchronized(i1) { // y1
   Integer i2 = Arrays.asList(6,7,8,9,10)
      .parallelStream()
      .sorted()  // y2
      .findAny().get(); // y3
   System.out.println(i1+" "+i2);
}
```

A. It outputs 1 6.
B. It outputs 1 10.
C. The code will not compile because of line y1.
D. The code will not compile because of line y2.
E. The code will not compile because of line y3.
F. It compiles but throws an exception at runtime.
G. The output cannot be determined ahead of time.
H. It compiles but waits forever at runtime.

59. Assuming / is the root directory, which of the following are true statements? (Choose all that apply.)

A. /home/parrot is an absolute path.
B. /home/parrot is a directory.
C. /home/parrot is a relative path.
D. The path pointed to from a File object must exist.
E. The parent of the path pointed to by a File object must exist.

60. What is the output of the following code?

```
Path path1 = Paths.get("/pets/../cat.txt");
Path path2 = Paths.get("./dog.txt");
System.out.println(path1.resolve(path2));
System.out.println(path2.resolve(path1));
```

A. /pets/../cat.txt/./dog.txt
   /pets/../cat.txt

Z komentarzem [57]: G. The code compiles and runs without issue, so C, D, E, F, and H are incorrect. There are two important things to notice: first, synchronizing on the first output doesn't actually impact the results of the code. Second, sorting on a parallel stream does not mean that findAny() will return the first record. The findAny() method will return the value from the first thread that retrieves a record. Therefore, the output is not guaranteed for either serial or parallel stream. Since the results cannot be predicted ahead of time, G is the correct answer.

Z komentarzem [58]: A. Paths that begin with the root directory are absolute paths, so A is correct and C is incorrect. B is incorrect because the path could be a file or directory within the file system. A File object may refer to a path that does not exist within the file system, so D and E are incorrect.

Z komentarzem [59]: A. The code compiles and runs without issue, so E is incorrect. For this question, you have to remember two things. First, the resolve() method does not normalize any path symbols, so C and D are not correct. Second, calling resolve() with an absolute path as a parameter returns the absolute path, so A is correct and B is incorrect.

B. /pets/../cat.txt/./dog.txt
./dog.txt/pets/../cat.txt
C. /cats.txt
/dog.txt
D. /cats.txt/dog.txt
/cat.txt
E. It compiles but throws an exception at runtime.

61. What is the result of the following code?

```
1:    public class Outer {
2:    private int x = 24;
3:    public int getX() {
4:       String message = "x is ";
5:       class Inner {
6:          private int x = Outer.this.x;
7:          public void printX() {
8:             System.out.println(message + x);
9:          }
10:      }
11:      Inner in = new Inner();
12:      in.printX();
13:      return x;
14:   }
15:   public static void main(String[] args) {
16:      new Outer().getX();
17:   } }
```

A. x is 0.
B. x is 24.
C. Line 6 generates a compiler error.
D. Line 8 generates a compiler error.
E. Line 11 generates a compiler error.
F. An exception is thrown.

62. Which of the answer choices are valid given the following declaration?

```
Map<String, Double> map = new HashMap<>();
```

A. map.add("pi", 3.14159);
B. map.add("e", 2L);
C. map.add("log(1)", new Double(0.0));
D. map.add('x', new Double(123.4));
E. None of the above

83. What changes need to be made to make the following singleton pattern correct? (Choose all that apply.)

Z komentarzem [60]: B. Outer.this.x is the correct way to refer to x in the Outer class. In Java 7, the answer would have been D because you used to have to declare variables as final to use them in a local inner class. In Java 8, this requirement was dropped and the variables only need to be effectively final, which means that the code would still compile if final were added.

Z komentarzem [61]: E. Trick question! The Map interface uses put() rather than add() to add elements to the map. If these examples used put(), the answer would be A and C. B is no good because a long cannot be shoved into a Double. D is no good because a char is not the same thing as a String.

Z komentarzem [62]: D, F. A is incorrect, as there are definitely some problems with the singleton implementation. B and C are incorrect, as naming of the instance variable and access method are not required as part of the pattern. The public modifier on the cheetahManager instance means that any class can access or even replace the instance, which breaks the singleton pattern; hence D is required to fix the implementation. E is incorrect, as marking the instance final would prevent lazy instantiation and as the code would not compile. F is also required, since without this step two threads could create two distinct instances of the singleton at the same time, which would violate the singleton pattern.

```
public class CheetahManager {
   public static CheetahManager cheetahManager;
   private CheetahManager() {}
   public static CheetahManager getCheetahManager() {
      if(cheetahManager == null) {
         cheetahManager = new CheetahManager();
      }
      return cheetahManager;
   }
}
```

A. None; the singleton pattern is properly implemented.
B. Rename cheetahManager to instance.
C. Rename getCheetahManager() to getInstance().
D. Change the access modifier of cheetahManager from public to private.
E. Mark cheetahManager final.
F. Add synchronized to getCheetahManager().

63. Which of the following can we add after line 5 for the code to run without error and not produce any output? (Choose all that apply.)

```
4: LongStream ls = LongStream.of(1, 2, 3);
5: OptionalLong opt = ls.map(n -> n * 10)
      .filter(n -> n < 5).findFirst();
```

A. if (opt.isPresent()) System.out.println(opt.get());
B. if (opt.isPresent()) System.out.println(opt.getAsLong());
C. opt.ifPresent(System.out.println)
D. opt.ifPresent(System.out::println)
E. None of these; the code does not compile.
F. None of these; line 5 throws an exception at runtime.

Z komentarzem [63]: B, D. Option A would work for a regular Stream. However, we have a LongStream and therefore need to call getAsLong(). Option C is missing the :: that would make it a method reference. Therefore, options B and D are correct.

64. Which of the following prints OhNo with the assertion failure when the number is negative? (Choose all that apply.)

A. assert n < 0: "OhNo";
B. assert n < 0, "OhNo";
C. assert n < 0 ("OhNo");
D. assert(n < 0): "OhNo";
E. assert(n < 0, "OhNo");

Z komentarzem [64]: A, D. An assertion consists of a boolean expression followed by an optional colon and message. The boolean expression is allowed to be in parenthesis, but this is not required. Therefore A and D are correct.

65. What is the output of the following code?

```
LocalDate date = LocalDate.of(2018, Month.APRIL, 30);
date.plusDays(2);
date.plusYears(3);
System.out.println(date.getYear() + " "
   + date.getMonth() + " "+ date.getDayOfMonth());
```

A. 2018 APRIL 2
B. 2018 APRIL 30
C. 2018 MAY 2
D. 2021 APRIL 2
E. 2021 APRIL 30
F. 2021 MAY 2
G. A runtime exception is thrown.

Z komentarzem [65]: B. The date starts out as April 30, 2018. Since dates are immutable and the plus methods have their return values ignored, the result is unchanged. Therefore, Option B is correct.

66. What are the requirements for a class that you want to serialize with ObjectOutputStream? (Choose all that apply.)

A. The class must implement the Serializable interface.
B. The class must extend the Serializable class.
C. The class must declare a static serialVersionUID variable.
D. All instance members of the class must be Serializable.
E. All instance members of the class must be marked transient.
F. Any class can be serialized with ObjectOutputStream.

Z komentarzem [66]: A. First, the class must implement the Serializable interface, so A is correct. Serializable is not a class; therefore B is incorrect. Creating a static serialVersionUID variable is optional and recommended, but it is not required for use with the ObjectOutputStream, so C is incorrect. Every instance variable must either be Serializable or be marked transient, but all variables are not required to be either, so D and E are incorrect. F is incorrect, because the class must be Serializable and have instance members that are Serializable or marked transient.

67. What are some advantages of using Files.lines() over Files.readAllLines()? (Choose all that apply.)

A. It is often faster.
B. It can be run on large files with very little memory available.
C. It can be chained with stream methods directly.
D. It does not modify the contents of the file.
E. It ensures the file is not read-locked by the file system.
F. There are no differences, because one method is a pointer to the other.

Z komentarzem [67]: B, C. The methods are not the same, because Files.lines() returns a Stream<Path> and Files.readAllLines() returns a List<String>, so F is incorrect. A is incorrect, because performance is not often the reason to prefer one to the other. Files.lines() reads the file in a lazy manner, while Files.readAllLines() reads the entire file into memory all at once; therefore Files.lines() works better on large files with limited memory available, and B is correct. Although a List can be converted to a stream with the stream() method, this requires an extra step; therefore C is correct since the resulting object can be chained directly to a stream. Finally, D and E are incorrect because they are not relevant to these methods.

68. Which of the options can fill in the blanks in order to make the code compile?

```
boolean bool = stmt._________(sql);
int num = stmt._________ (sql);
ResultSet rs = stmt._________ (sql);
```

A. execute, executeQuery, executeUpdate
B. execute, executeUpdate, executeQuery
C. executeQuery, execute, executeUpdate
D. executeQuery, executeUpdate, execute

Z komentarzem [68]: B. The first line has a return type of boolean because any type of SQL statement can be run, making it an execute() call. The second line returns the number of modified rows, making it an executeUpdate() call. The third line returns the results of a query, making it an executeQuery() call.

E. executeUpdate, execute, executeQuery
F. executeUpdate, executeQuery, execute

69. Assuming MyTask is an abstract class that implements the ForkJoinTask interface, what statements about the following code are true? (Choose all that apply.)

```
import java.util.concurrent.*;
public class FindMin extends MyTask {
   private Integer[] elements;
   private int a;
   private int b;
   public FindMin(Integer[] elements, int a, int b) {
      this.elements = elements;
      this.a = a;
      this.b = b;
   }
   public Integer compute() {
      if ((b—a) < 2)
         return Math.min(elements[a], elements[b]);
      else {
         int m = a + ((b—a) / 2);
         System.out.println(a + "," + m + "," + b);
         MyTask t1 = new FindMin(elements, a, m);
         int result = t1.fork().join();
         return Math.min(new FindMin(elements, m, b).compute(),
result);
      }
   }

   public static void main(String[] args) throws
InterruptedException,

ExecutionException {
      Integer[] elements = new Integer[] { 8, -3, 2, -54 };
      MyTask task = new FindMin(elements, 0, elements.length—1);
      ForkJoinPool pool = new ForkJoinPool(1);
      Integer sum = pool.invoke(task);
      System.out.println("Min: " + sum);
   }
}
```

A. The code correctly finds the minimum value in the array.
B. MyTask inherits RecursiveAction.
C. MyTask inherits RecursiveTask.
D. The code produces a ForkJoinPool at runtime.
E. The class produces single-threaded performance at runtime.
F. The code does not compile.

Z komentarzem [69]: A, C, E. The code compiles without issue, so F is incorrect. Note that the compute() method is protected in the parent class, although you can override it with public without issue since this is a more accessible modifier. Since compute() returns a value, RecursiveTask must be inherited instead of RecursiveAction, so C is correct and B is incorrect. The code does correctly find the minimum value on a non-empty array without entering an infinite loop, so A is correct and D is incorrect. Finally, since the code calls join() immediately after fork(), causing the process to wait, it does not perform any faster if there are 100 threads versus 1 thread, so E is also correct.