socrative

Answer Key
February 24, 2021

OCA Practice Qustions Ch. 8 ('Copy')

1.

```
package mind;
public class Remember {
   public static void think() throws Exception {  // k1
      try {
         throw new Exception();
      }
   }
   public static void main(String... ideas) throws Exception {
      think();
   }
}
```

What is the result of compiling and executing the following application?

- A The code compiles and runs without printing anything.
- B The code compiles but a stack trace is printed at runtime.
- C The code does not compile because of line k1.
- ✓ D The code does not compile for another reason.

i A try block must include either a catch or finally block, or both. The think() method declares a try block but neither additional block. For this reason, the code does not compile, and Option D is the correct answer. The rest of the lines compile without issue, including k1.

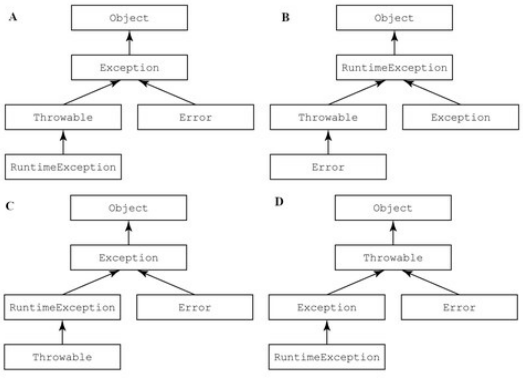
2. Choose the answer that lists the keywords in the order that they would be used together.

- A catch, try, finally
- ✓ B try, catch, finally
- C finally, catch, try
- D try, finally, catch

i The correct order of blocks is try, catch, and finally, making Option B the correct answer.

3. Which of the following diagrams of java.lang classes shows the inheritance model properly?

- A A
- B B
- C C
- ✓ D D



i Option D is the correct model. The class RuntimeException extends Exception, and both Exception and Error extend Throwable. Finally, like all Java classes, they all inherit from Object. Notice that Error does not extend Exception, even though we often refer to these generally as exceptions.

4. Which of the following Throwable types is it recommended not to catch in a Java application?

✓ **A** Error

B CheckedException

C Exception

D RuntimeException

i While Exception and RuntimeException are commonly caught in Java applications, it is recommended Error not be caught. An Error often indicates a failure of the JVM which cannot be recovered from. For this reason, Option A is correct, and Options C and D are incorrect. Option B is not a class defined in the Java API; therefore, it is also incorrect.

5. What is the output of the following application?

A 123

B 124

C 12

✓ **D** None of the above

```
package game;
public class Baseball {
   public static void main(String... teams) {
      try {
         int score = 1;
         System.out.print(score++);
      } catch (Throwable t) {
         System.out.print(score++);
      } finally {
         System.out.print(score++);
      }
      System.out.print(score++);
   }
}
```

i The application does not compile because score is defined only within the try block. The other three places it is referenced, in the catch block, in the finally block, and outside the try-catch-finally block at the end, are not in scope for this variable and each does not compile. Therefore, the correct answer is Option D.

6. Which of the following is a checked exception?

A ClassCastException

✓ **B** IOException

C ArrayIndexOutOfBoundsException

D IllegalArgumentException

i ClassCastException, ArrayIndexOutOfBoundsException, and IllegalArgumentException are unchecked exceptions and can be thrown at any time. IOException is a checked exception that must be handled or declared when used, making Option B the correct answer.

7. Fill in the blanks: The _____ keyword is used in method declarations, while the _____ keyword is used to throw an exception to the surrounding process.

✓ **A** throws, throw

B catch, throw

C throw, throws

D throws, catch

i The throws keyword is used in method declarations, while the throw keyword is used to throw an exception to the surrounding process, making Option A the correct answer. The catch keyword is used to

handle exceptions, not to create them or in the declaration of a method.

8. If a try statement has catch blocks for both Exception and IOException, then which of the following statements is correct?

A The catch block for Exception must appear before the catch block for IOException.

✓ B The catch block for IOException must appear before the catch block for Exception.

C The catch blocks for these two exception types can be declared in any order.

D A try statement cannot be declared with these two catch block types because they are incompatible.

i IOException is a subclass of Exception, so it must appear first in any related catch blocks. If Exception was to appear before IOException, then the IOException block would be considered unreachable code because any thrown IOException is already handled by the Exception catch block. For this reason, Option B is correct.

9.

```
package game;
public class Football {
   public static void main(String officials[]) {
      try {
         System.out.print('A');
         throw new RuntimeException("Out of bounds!");
      } catch (ArrayIndexOutOfBoundsException aioobe)
         System.out.print('B');
         throw aioobie;
      } finally {
         System.out.print('C');
      }
   }
}
```

What is the output of the following application?

A ABC

B ABC, followed by a stack trace for a RuntimeException

✓ C AC, followed by a stack trace for a RuntimeException

D None of the above

i The application first enters the try block and outputs A. It then throws a RuntimeException, but the exception is not caught by the catch block since RuntimeException is not a subclass of ArrayIndexOutOfBoundsException (it is a superclass). Next, the finally block is called and C is output. Finally, the RuntimeException is thrown by the main() method and a stack trace is printed. For these reasons, Option C is correct.

10.

```
package castles;
public class Fortress {
   public void openDrawbridge() throws Exception {  // p1
      try {
         throw new Exception("Circle");
      } catch (Exception e) {
         System.out.print("Opening!");
      } finally {
         System.out.print("Walls");  // p2
      }
   }
   public static void main(String[] moat) {
      new Fortress().openDrawbridge();  // p3
   }
}
```

What is the result of compiling and running the following application?

A The code does not compile because of line p1.

B The code does not compile because of line p2.

✓ C The code does not compile because of line p3.

D The code compiles, but a stack trace is printed at runtime.

i The application does not compile, so Option D is incorrect. The openDrawbridge() method compiles without issue, so Options A and B are incorrect. The issue here is how the openDrawbridge() method is called from within the main() method on line p3. The openDrawbridge() method declares the checked exception, Exception, but the main() method from which it is called does not handle or declare the exception. In order for this code to compile, the main() method would have to have a try-catch statement around line p3 that properly handles the checked exception, or the main() would have to be updated to declare a compatible checked exception. For these reasons, line p3 does not compile, and Option C is the correct answer.

11. Which of the following exception types must be handled or declared by the method in which they are thrown?

A NullPointerException

✓ B Exception

C RuntimeException

D ArithmeticException

i NullPointerException and ArithmeticException both extend RuntimeException, which are unchecked exceptions and not required to be handled or declared in the method in which they are thrown. On the other hand, Exception is a checked exception and must be handled or declared by the method in which it is thrown. Therefore, Option B is the correct answer.

12. What is the output of the following application?

✓ A 1345

B 1235

C The code does not compile.

D The code compiles but throws an exception at runtime.

```
package game;
public class BasketBall {
   public static void main(String[] dribble) {
      try {
         System.out.print(1);
         throw new ClassCastException();
      } catch (ArrayIndexOutOfBoundsException ex) {
         System.out.print(2);
      } catch (Throwable ex) {
         System.out.print(3);
      } finally {
         System.out.print(4);
      }
      System.out.print(5);
   }
}
```

i The code compiles and runs without issues, so Options C and D are incorrect. The try block throws a ClassCastException. Since ClassCastException is not a subclass of ArrayIndexOutOfBoundsException,

the first catch block is skipped. For the second catch block, ClassCastException is a subclass of Throwable, so that block is executed. Afterward, the finally block is executed and then control returns to the main() method with no exception being thrown. The result is that 1345 is printed, making Option A the correct answer.

13. Which of the following statements about a finally block is true?

A Every line of the finally block is guaranteed to be executed.

B The finally block is executed only if the related catch block is also executed.

✓ C The finally statement requires brackets {}.

D The finally block cannot throw an exception.

i A finally block can throw an exception, in which case not every line of the finally block would be executed. For this reason, Options A and D are incorrect. Option B is also incorrect The finally block is called regardless of whether or not the related catch block is executed. Option C is the correct answer. Unlike an if-then statement, which can take a single statement, a finally statement requires brackets {}.

14. Given that FileNotFoundException is a subclass of IOException, what is the output of the following application?

```
package office;
import java.io.*;
public class Printer {
   public void print() {
      try {
         throw new FileNotFoundException();
      } catch (IOException exception) {
         System.out.print("Z");
      } catch (FileNotFoundException enfe) {
         System.out.print("X");
      } finally {
         System.out.print("Y");
      }
   }
   public static void main(String... ink) {
      new Printer().print();
   }
}
```

A XY

B ZY

✓ C The code does not compile.

D The code compiles but a stack trace is printed at runtime.

i The code does not compile because the catch blocks are used in the wrong order. Since IOException is a superclass of FileNotFoundException, the FileNotFoundException is considered unreachable code. For this reason, the code does not compile, and Option C is correct.

15. Which keywords are required with a try statement?
I. catch
II. finalize
III. finally

A I only

B II only

✓ C I or III, or both

D None of these statements are required with a try statement.

i A try statement requires a catch or a finally block. Without one of them, the code will not compile; therefore, Option D is incorrect. A try statement can also be used with both a catch and finally block, making Option C the correct answer. Note that finalize is not a keyword, but a method inherited from java.lang.Object.

16. Which statement about the role of exceptions in Java is incorrect?

A Exceptions are often used when things “go wrong” or deviate from the expected path.

✓ B An application that throws an exception will terminate.

C Some exceptions can be avoided programmatically.

D An application that can properly handle its exception may recover from unexpected problems.

i Option A is a true statement about exceptions and when they are often applied. Option B is the false statement and the correct answer. An application that throws an exception can choose to handle the exception and avoid termination. Option C is also a true statement. For example, a NullPointerException can be avoided on a null object by testing whether or not the object is null before attempting to use it. Option D is also a correct statement. Attempting to recover from unexpected problems is an important aspect of proper exception handling.

17.

```
package harbor;
class CapsizedException extends Exception {}
class Transport {
   public int travel() throws CapsizedException { return 2; };
}
public class Boat {
   public int travel() throws Exception { return 4; };  // j1
   public static void main(String... distance) throws Exception{
      try {
         System.out.print(new Boat().travel());
      } catch (Exception e) (
         System.out.print(8);
      )
   }
}
```

What is the output of the following application?

A 4

B 8

C The code does not compile due to line j1.

✓ D The code does not compile for another reason.

i The code does not compile because the catch block uses parentheses () instead of brackets {}, making Option D the correct answer. Note that Boat does not extend Transport, so while the override on line j1 appears to be invalid since Exception is a broader checked exception than CapsizedException, that code compiles without issue. If the catch block was fixed, the code would output 4, making Option A the correct answer.

18.

```
class PrintException extends Exception {}
class PaperPrintException extends PrintException {}

public interface Printer {
   abstract int printData() throws PrintException;
}
```

Which of following method signatures would not be allowed in a class implementing the Printer interface?

A public int printData() throws PaperPrintException

✓ B public int printData() throws Exception

C public int printData()

D None of the above

i Overridden methods cannot throw new or broader checked exceptions than the one they inherit. Since Exception is a broader checked exception than PrintException, Option B is not allowed and is the correct choice. Alternatively, declaring narrower or the same checked exceptions or removing them entirely is allowed, making Options A and C incorrect. Since Option B is correct, Option D is incorrect.

19. Which import statement is required to be declared in order to use the Exception, RuntimeException, and Throwable classes in an application?

A import java.exception.*;

B import java.util.exception.*;

C import java.lang.*;

✓ D None of the above

i All three of those classes belong to the java.lang package, so Option C seems like the correct answer. The Java compiler, though, includes java.lang by default, so no import statement is actually required to use those three classes, making Option D the correct answer.

20.

```
class GasException extends Exception {}
class Element {
   public int getSymbol() throws GasException { return -1; }  // g1
}
public class Oxygen extends Element {
   public int getSymbol() { return 8; }  // g2
   public void printData() {
      try {
         System.out.print(getSymbol());
      } catch {  // g3
         System.out.print("Unable to read data");
      }
   }
}
```

Which statement about the following classes is correct?

A The code does not compile because of line g1.

B The code does not compile because of line g2.

✓ C The code does not compile because of line g3.

D None of the above

i The code does not compile because the catch block is missing a variable type and name, such as catch (Exception e). Therefore, Option C is the correct answer. Both implementations of getSymbol() compile without issue, including the overridden method. A subclass can swallow a checked exception for a method declared in a parent class; it just cannot declare any new or broader checked exceptions.

21. Fill in the blanks: A program must handle or declare _____ but should never handle _____.

A java.lang.Error, unchecked exceptions

✓ B checked exceptions, java.lang.Error

C java.lang.Throwable, java.lang.Error

D unchecked exceptions, java.lang.Exception

i Checked exceptions must be handled or declared or the program will not compile, while unchecked exceptions can be optionally handled. On the other hand, java.lang.Error should never be handled by the

application because it often indicates an unrecoverable state in the JVM, such as running out of memory. For these reasons, Option B is the correct answer.

22.

```
package castles;
class CastleUnderSiegeException extends Exception {}
class KnightAttackingException extends CastleUnderSiegeException {}
public class Citadel {
   public void openDrawbridge() throws RuntimeException {  // q1
      try {
         throw new KnightAttackingException();
      } catch (Exception e) {
         throw new ClassCastException();
      } finally {
         throw new CastleUnderSiegeException();  // q2
      }
   }
   public static void main(String[] moat) {
      new Citadel().openDrawbridge();  // q3
   }
}
```

What is the result of compiling and running the following application?

A The code does not compile because of line q1.

✓ B The code does not compile because of line q2.

C The code does not compile because of line q3.

D The code compiles, but a stack trace is printed at runtime.

i The application does not compile, so Option D is incorrect. The checked KnightAttackingException thrown in the try block is handled by the associated catch block. The ClassCastException is an unchecked exception, so it is not required to be handled or declared and line q1 compiles without issue. The finally block throws a checked CastleUnderSiegeException, which is required to be handled or declared by the method, but is not. There is no try-catch around line q2, and the method does not declare a compatible checked exception, only an unchecked exception. For this reason, line q2 does not compile, and Option B is the correct answer. Lastly, line q3 compiles without issue because the unchecked RuntimeException is not required to be handled or declared by the call in the main() method.

23. If an exception matches two or more catch blocks, which catch block is executed?

✓ A The first one that matches is executed.

B The last one that matches is executed.

C All matched blocks are executed.

D It is not possible to write code like this.

i If an exception matches multiple catch blocks, the first one that it encounters will be the only one executed, making Option A correct, and Options B and C incorrect. Option D is also incorrect. It is possible to write two consecutive catch blocks that can catch the same exception, with the first type being a subclass of the second. In this scenario, an exception thrown of the first type would match both catch blocks, but only the first catch block would be executed, since it is the more specific match.

24. What is the output of the following application?

```
package system;
public class Computer {
   public void compute() throws Exception {
      throw new RuntimeException("Error processing request");
   }
   public static void main(String[] bits) {
      try {
         new Computer().compute();
         System.out.print("Ping");
      } catch (NullPointerException e) {
         System.out.print("Pong");
         throw e;
      }
   }
}
```

A Ping

B Pong

✓ C The code does not compile.

D The code compiles but throws an exception at runtime.

i The code does not compile due to the call to compute() in the main() method. Even though the compute() method only throws an unchecked exception, its method declaration includes the Exception class, which is a checked exception. For this reason, the checked exception must be handled or declared in the main() method in which it is called. While there is a try-catch block in the main() method, it is only for the unchecked NullPointerException. Since Exception is not a subclass of NullPointerException, the checked Exception is not properly handled or declared and the code does not compile, making Option C the correct answer.

25. In the following application, the value of list has been omitted. Assuming the code compiles without issue, which one of the following is not a possible output of executing this class?

```
package checkboard;

public class Attendance {
   private Boolean[] list = // value omitted
   public int printTodaysCount() {
      int count=0;
      for(int i=0; i<10; i++) {
         if(list[i]) ++count;
      }
      return count;
   }
   public static void main(String[] roster) {
      new Attendance().printTodaysCount();
   }
}
```

A A stack trace for NullPointerException is printed.

B A stack trace for ArrayIndexOutOfBoundsException is printed.

C A stack trace for ClassCastException is printed.

✓ D None of the above

i A NullPointerException can be thrown if the value of list is null. Likewise, an ArrayIndexOutOfBoundsException can be thrown if the value of list is an array with fewer than 10 elements. Finally, a ClassCastException can be thrown if list is assigned an object that is not of type Boolean[]. For example, the assignment list = (Boolean[]) new Object() will compile without issue but throws a ClassCastException at runtime. Therefore, the first three options are possible, making Option D the correct answer.

26. Fill in the blanks: A _____ occurs when a program recurses too deeply into an infinite loop, while a(n) _____ occurs when a reference to a nonexistent object is acted upon.

A NoClassDefFoundError, StackOverflowError

✓ B StackOverflowError, NullPointerException

C ClassCastException, IllegalArgumentException

D StackOverflowError, IllegalArgumentException

i A StackOverflowError occurs when a program recurses too deeply into an infinite loop. It is considered an error because the JVM often runs out of memory and cannot recover. A NullPointerException occurs when an instance method or variable on a null reference is used. For these reasons, Option B is correct. A NoClassDefFoundError occurs when code available at compile time is not available at runtime. A

ClassCastException occurs when an object is cast to an incompatible reference type. Finally, an IllegalArgumentException occurs when invalid parameters are sent to a method.

27. Which of the following is not a reason to add checked exceptions to a method signature?

- **A** To force a caller to handle or declare its exceptions
- **B** To notify the caller of potential types of problems
- ✓ **C** To ensure that exceptions never cause the application to terminate
- **D** To give the caller a chance to recover from a problem

i Checked exceptions are commonly used to force a caller to deal with an expected type of problem, such as the inability to write a file to the file system. Without dealing with all checked exceptions thrown by the method, the calling code does not compile, so Option A is a true statement. Option B is also a true statement. Declaring various different exceptions informs the caller of the potential types of problems the method can encounter. Option C is the correct answer. There may be no recourse in handling an exception other than to terminate the application. Finally, Option D is also a true statement because it gives the caller a chance to recover from an exception, such as writing file data to a backup location.

28.

```
package peculiar;
public class Stranger {
   public static String getFullName(String firstName, String lastName) {
      try {
         return firstName.toString() + " " + lastName.toString();
      } finally {
         System.out.print("Finished!");
      } catch (NullPointerException npe) {
         System.out.print("Problem?");
      }
      return null;
   }
   public static void main(String[] things) {
      System.out.print(getFullName("Joyce","Hopper"));
   }
}
```

What is the output of the following application?

- **A** Joyce Hopper
- **B** Finished!Joyce Hopper
- **C** Problem?Finished!null
- **D** None of the above

i This code does not compile because the catch and finally blocks are in the wrong order, making Option D the correct answer. If the order was flipped, the output would be Finished!Joyce Hopper, making Option B correct.

29. Fill in the blanks: A try statement has _____ finally block(s) and _____ catch blocks.

- ✓ **A** zero or one, zero or more
- **B** one, one or more
- **C** zero or one, zero or one
- **D** one or more, zero or one

i A try statement is not required to have a finally block, but if it does, there can be at most one. Furthermore, a try statement can have any number of catch blocks or none at all. For these reasons, Option A is the correct answer.

30.

```
package pond;
abstract class Duck {
   protected int count;
   public abstract int getDuckies();
}
public class Ducklings extends Duck {
   private int age;
   public Ducklings(int age) { this.age = age; }
   public int getDuckies() { return this.age/count; }
   public static void main(String[] pondInfo) {
      Duck itQuacks = new Ducklings(5);
      System.out.print(itQuacks.getDuckies());
   }
}
```

What is the output of the following application?

A 0

B 5

C The code does not compile.

✓ D The code compiles but throws an exception at runtime.

i The code compiles without issue, so Option C is incorrect. The key here is noticing that count, an instance variable, is initialized with a value of 0. The getDuckies() method ends up computing 5/0, which leads to an unchecked ArithmeticException at runtime, making Option D the correct answer.

31. Given a try statement, if both the catch block and the finally block each throw an exception, what does the caller see?

A The exception from the catch block

✓ B The exception from the finally block

C Both the exception from the catch block and the exception from the finally block

D None of the above

i If both the catch and finally blocks throw an exception, the one from the finally block is propagated to the caller, with the one from the catch block being dropped, making Option B the correct answer. Note that Option C is incorrect due to the fact that only one exception can be thrown to the caller.

32.

```
package zoo;
class BigCat {
   void roar(int level) throw RuntimeException {  // m1
      if(level<3) throw new IllegalArgumentException("Incomplete");
      System.out.print("Roar!");
   }
}
public class Lion extends BigCat {
   public void roar() {  // m2
      System.out.print("Roar!!!");
   }

   public static void main(String[] cubs) {
      final BigCat kitty = new Lion();  // m3
      kitty.roar(2);
   }
}
```

What is the output of the following application?

✓ A The code does not compile because of line m1.

B The code does not compile because of line m2.

C The code does not compile because of line m4.

D The code compiles but a stack trace is printed at runtime.

i The application does not compile because the roar() method in the BigCat class uses throw instead of

throws, making Option A the correct answer. Note that if the correct keyword was used, the code would compile without issue, and Option D would be correct. Also the override of roar() in the Lion class is valid, since the overridden method has a broader access modifier and does not declare any new or broader checked exceptions.

33.

```
final Object exception = new Exception();
final Exception data = (RuntimeException)exception;
System.out.print(data);
```

Given the following code snippet, which specific exception will be thrown?

✓ **A** ClassCastException

B RuntimeException

C NullPointerException

D None of the above

i Although this code uses the RuntimeException and Exception classes, the question is about casting. Exception is not a subclass of RuntimeException, so the assignment on the second line throws a ClassCastException at runtime, making Option A correct.

34. Which of the following classes will handle all types in a catch block?

A Exception

B Error

✓ **C** Throwable

D RuntimeException

i All exceptions in Java inherit from Throwable, making Option C the correct answer. Note that Error and Exception extend Throwable, and RuntimeException extends Exception.

35.

```
package registration;
public class Address {
   public String getAddress(String street, String city) {
      try {
         return street.toString() + " - " + city.toString();
      } finally {
         System.out.print("Posted:");
      }
   }

   public static void main(String[] form) {
      String street = // value omitted
      String city = // value omitted
      System.out.print(new Address().getAddress(street,city));
   }
}
```

In the following application, the values of street and city have been omitted. Which one of the following is a possible output of executing this class?
I. 350 5th Ave - New York
II. Posted:350 5th Ave - New York

A I only

✓ **B** II only

C I and II

D None of the above

i If both values are valid non-null String objects, then no exception will be thrown, with the statement in the finally block being executed first, before returning control to the main() method; therefore, the second statement is a possible output. If either value is null, then the toString() method will cause a NullPointerException to be thrown. In both cases, the finally block will execute first, printing Posted:, even if there is an exception. For this reason, the first statement is not a possible output, and Option B is correct.

36. If a try statement has catch blocks for both ClassCastException and RuntimeException, then which of the following statements is correct?

✓ **A** The catch block for ClassCastException must appear before the catch block for RuntimeException.

B The catch block for RuntimeException must appear before the catch block for ClassCastException.

C The catch blocks for these two exception types can be declared in any order.

D A try statement cannot be declared with these two catch block types because they are incompatible.

i ClassCastException is a subclass of RuntimeException, so it must appear first in any related catch blocks. If RuntimeException was to appear before ClassCastException, then the ClassCastException block would be considered unreachable code, since any thrown ClassCastException is already handled by the RuntimeException catch block. For this reason, Option A is correct.

37. Which of the following is the best scenario to use an exception?

A The computer caught fire.

B The code does not compile.

✓ **C** A caller passes invalid data to a method.

D A method finishes sooner than expected.

i Option A is incorrect. You should probably seek help if the computer is on fire! Option B is incorrect because code that does not compile cannot run and therefore cannot throw any exceptions. Option C is the best answer, since an IllegalArgumentException can be used to alert a caller of missing or invalid data. Option D is incorrect; finishing sooner is rarely considered a problem.

38.

```
package body;
class Organ {
   public void operate() throws RuntimeException {
      throw new RuntimeException("Not supported");
   }
}
public class Heart extends Organ {
   public void operate() throws Exception {
      System.out.print("beat");
   }
   public static void main(String... cholesterol) throws Exception {
      try {
         new Heart().operate();
      } finally {
      }
   }
}
```

What is the output of the following application?

A beat

B Not supported

✓ C The code does not compile.

D The code compiles but a stack trace is printed at runtime.

i The code does not compile due to an invalid override of the operate() method. An overridden method must not throw any new or broader checked exceptions than the method it inherits. Even though RuntimeException is a subclass of Exception, Exception is considered a new checked exception, since RuntimeException is an unchecked exception. Therefore, the code does not compile, and Option C is correct.

39.

```
throw new NullPointerException();
```

Which statement about the following exception statement is correct?

A The code where this is called must include a try-catch block that handles this exception.

B The method where this is called must declare a compatible exception.

C This exception cannot be handled.

✓ D This exception can be handled with a try-catch block or ignored altogether by the surrounding method.

i A NullPointerException is an unchecked exception. While it can be handled by the surrounding method, either through a try-catch block or included in the method declaration, these are optional. For this reason, Option D is correct.

40. What is the output of the following application?

A Finished!

B Finished!, followed by a stack trace

C The application does not produce any output at runtime.

✓ D The code does not compile.

```
package clothing;
public class Coat {
   public Long zipper() throws Exception {
      try {
         String checkZipper = (String)new Object();
      } catch (Exception e) {
         throw RuntimeException("Broken!");
      }
      return null;
   }
   public static void main(String... warmth) {
      try {
         new Coat().zipper();
         System.out.print("Finished!");
      } catch (Throwable t) {}
   }
}
```

i In this application, the throw RuntimeException(String) statement in the zipper() method does not include the new keyword. The new keyword is required to create the object being thrown, since RuntimeException(String) is a constructor. For this reason, the code does not compile, and Option D is correct. If the keyword new was inserted properly, then the try block would throw a ClassCastException,

which would be replaced with a RuntimeException to the calling method by the catch block. The catch block in the main() method would then be activated, and no output would be printed, making Option C correct.

41. Given the following application, which type of exception will be printed in the stack trace at runtime?

```
package carnival;
public class WhackAnException {
   public static void main(String... hammer) {
      try {
         throw new ClassCastException();
      } catch (IllegalArgumentException e) {
         throw new IllegalArgumentException();
      } catch (RuntimeException e) {
         throw new NullPointerException();
      } finally {
         throw new RuntimeException();
      }
   }
}
```

A IllegalArgumentException

B NullPointerException

✓ C RuntimeException

D The code does not compile.

i For this question, notice that all the exceptions thrown or caught are unchecked exceptions. First, the ClassCastException is thrown in the try block and caught by the second catch block since it inherits from RuntimeException, not IllegalArgumentException. Next, a NullPointerException is thrown, but before it can be returned the finally block is executed and a RuntimeException replaces it. The application exits and the caller sees the RuntimeException in the stack trace, making Option C the correct answer. If the finally block did not throw any exceptions, then Option B would be the correct answer.

42.

```
class OutOfBoundsException extends BadCatchException {}
class BadCatchException extends Exception {}

public interface Outfielder {
   public void catchBall() throws OutOfBoundsException;
}
```

Which of these method signatures is allowed in a class implementing the Outfielder interface?

A public int catchBall() throws OutOfBoundsException

B public int catchBall() throws BadCatchException

C public int catchBall() throws Exception

✓ D None of the above

i Trick question! Options A, B, and C are each invalid overrides of the method because the return type must be covariant with void. For this reason, Option D is the correct answer. If the return types were changed to be void, then Option A would be a valid override. Options B and C would still be incorrect, since overridden methods cannot throw broader checked exceptions than the inherited method.

43.

```
package city;
public class Street {
   public static void dancing() throws RuntimeException {
      try {
         throw new IllegalArgumentException();
      } catch (Error) {
         System.out.print("Unable!");
      }
   }
   public static void main(String... count) throws RuntimeException {
      dancing();
   }
}
```

What is the output of the following application?

A Unable!

B The application does not produce any output.

C The application compiles but produces a stack trace at runtime.

✓ D The code does not compile.

i The code does not compile because the catch block is missing a variable name, such as catch (Error e). Therefore, Option D is the correct answer. If a variable name was added, the application would produce a stack trace at runtime and Option C would be the correct answer. Because IllegalArgumentException does not inherit from Error, the catch block would be skipped and the exception sent to the main() method at runtime. Note that the declaration of RuntimeException by both methods is unnecessary since it is unchecked, although allowed by the compiler.

44.

```
package castles;
class DragonException extends Exception {}
public class Lair {
   public void openDrawbridge() throws Exception {  // r1
      try {
         throw new Exception("This Exception");
      } catch (RuntimeException e) {
         throw new DragonException();  // r2
      } finally {
         throw new RuntimeException("Or maybe this one");
      }
   }
   public static void main(String[] moat) throws Exception {
      new Lair().openDrawbridge();  // r3
   }
}
```

What is the result of compiling and running the following application?

A The code does not compile because of line r1.

B The code does not compile because of line r2.

C The code does not compile because of line r3.

✓ D The code compiles, but a stack trace is printed at runtime.

i The openDrawbridge() is capable of throwing a variety of exceptions, including checked Exception and DragonException as well as an unchecked RuntimeException. All of these are handled by the fact that the method declares the checked Exception class in the method signature, which all the exceptions within the class inherit. For this reason, the openDrawbridge() method compiles without issue. The call to openDrawbridge() in the main() method also compiles without issue because the main() method declares Exception in its signature. For these reasons, the code compiles but a stack trace is printed at runtime, making Option D the correct answer. In case you are wondering, the caller would see RuntimeException: Or maybe this one in the stack trace at runtime, since the exception in the finally block replaces the one from the try block. Note that the exception in the catch block is never reached because the RuntimeException type declared in the catch block does not handle Exception.

45. If a try statement has catch blocks for both IllegalArgumentException and ClassCastException, then which of the following statements is correct?

A The catch block for IllegalArgumentException must appear before the catch block for ClassCastException.

B The catch block for ClassCastException must appear before the catch block for IllegalArgumentException.

✓ C The catch blocks for these two exception types can be declared in any order.

D A try statement cannot be declared with these two catch block types because they are incompatible.

i Both IllegalArgumentException and ClassCastException inherit RuntimeException, but neither is a subclass of the other. For this reason, they can be listed in either order, making Option C the correct statement.

46.

What is the output of the following application?

```
package broken;
class Problem implements RuntimeException {}
public class BiggerProblem extends Problem {
   public static void main(String uhOh[]) {
      try {
         throw new BiggerProblem();
      } catch (BiggerProblem re) {
         System.out.print("Problem?");
      } catch (Problem e) {
         System.out.print("Handled");
      } finally {
         System.out.print("Fixed!");
      }
   }
}
```

A Problem?Fixed!

B Handled.Fixed!

C Problem?Handled.Fixed!

✓ D The code does not compile.

i The class RuntimeException is not an interface and it cannot be implemented. For this reason, the Problem class does not compile, and Option D is the correct answer. Note that this is the only compilation problem in the application. If implements was changed to extends, the code would compile and Problem? Fixed! would be printed, making Option A the correct answer.

47.

```
package lighting;
interface Source {
   void flipSwitch() throws Exception;
}
public class LightBulb implements Source {
   public void flipSwitch() {
      try {
         throws new RuntimeException("Circuit Break!");
      } finally {
         System.out.print("Flipped!");
      }
   }
   public static void main(String... electricity) throws Throwable {
      final Source bulb = new LightBulb();
      bulb.flipSwitch();
   }
}
```

What is the output of the following application?

A A stack trace for a RuntimeException

B Flipped!, followed by a stack trace for a RuntimeException

C The code does not compile because flipSwitch() is an invalid method override.

✓ D The code does not compile for another reason.

i The question is designed to see how closely you pay attention to throw and throws! The try block uses the incorrect keyword, throws, to create an exception. For this reason, the code does not compile, and Option D is correct. If throws was changed to throw, then the code would compile without issue, and Option B would be correct.

48. Given an application that hosts a website, which of the following would most likely result in a java.lang.Error being thrown?

- **A** Two users try to register an account at the same time.
- **B** The application temporarily loses connection to the network.
- **C** A user enters their password incorrectly.
- ✓ **D** The application runs out of memory.

i A Java application tends to only throw an Error when the application has entered a final, unrecoverable state. Options A and C are incorrect. These types of errors are common and expected in most software applications, and should not cause the application to terminate. Option B uses the word *temporarily*, meaning the network connection will come back. In this case, a regular exception could be used to try to recover from this state. Option D is the correct answer because running out of memory is usually unrecoverable in Java.

49.

```
package storage;
import java.io.*;
public class Backup {
   public void performBackup() {
      try {
         throw new IOException("Disk not found");
      } catch (Exception e) {
         try {
            throw new FileNotFoundException("File not found");
         } catch (FileNotFoundException e) {  // z1
            System.out.print("Failed");
         }
      }
   }
   public static void main(String... files) {
      new Backup().performBackup();  // z2
   }
}
```

Given that FileNotFoundException is a subclass of IOException, what is the output of the following application?

- **A** Failed
- **B** The application compiles but a stack trace is printed at runtime.
- ✓ **C** The code does not compile because of line z1.
- **D** The code does not compile because of line z2.

i While a catch block is permitted to include an embedded try-catch block, the issue here is that the variable name e is already used by the first catch block. In the second catch block, it is equivalent to declaring a variable e twice. For this reason, line z1 does not compile, and Option C is the correct answer. If a different variable name was used for either catch block, then the code would compile without issue, and Option A would be the correct answer.

50. What is the output of the following application?

```
package bed;
public class Sleep {
   public static void snore() {
      try {
         String sheep[] = new String[3];
         System.out.print(sheep[3]);
      } catch (RuntimeException e) {
         System.out.print("Awake!");
      } finally {
         throw new Exception();  // x1
      }
   }
   public static void main(String... sheep) {  // x2
      new Sleep().snore();  // x3
   }
}
```

A Awake!, followed by a stack trace

✓ B The code does not compile because of line x1.

C The code does not compile because of line x2.

D The code does not compile because of line x3.

i The finally block of the snore() method throws a new checked exception on line x1, but there is no try-catch block around it to handle it, nor does the snore() method declare any checked exceptions. For these reasons, line x1 does not compile, and Option B is the correct answer. The rest of the lines of code compile without issue, even line x3 where a static method is being accessed using an instance reference. Note that the code inside the try block, if it ran, would produce an ArrayIndexOutOfBoundsException, which would be caught by the RuntimeException catch block, printing Awake!. What happens next would depend on how the finally block was corrected.