socrative

Answer Key
February 24, 2021

OCA Chapter 6 Review ('Copy')

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

A Runtime exceptions are the same thing as checked exceptions.

✓ B Runtime exceptions are the same thing as unchecked exceptions.

C You can declare only checked exceptions.

D You can declare only unchecked exceptions.

E You can handle only Exception subclasses.

i Runtime exceptions are also known as unchecked exceptions. They are allowed to be declared, but they don't have to be. Checked exceptions must be handled or declared. Legally, you can handle java.lang.Error subclasses, but it's not a good idea.

2. Which of the following pairs fill in the blanks to make this code compile? (Choose all that apply)

```
7: public void ohNo() ____ Exception {
8: ___________ Exception();
9: }
```

A On line 7, fill in throw

✓ B On line 7, fill in throws

C On line 8, fill in throw

✓ D On line 8, fill in throw new

E On line 8, fill in throws

F On line 8, fill in throws new

i In a method declaration, the keyword throws is used. To actually throw an exception, the keyword throw is used and a new exception is created.

3. When are you required to use a finally block in a regular try statement (not a try-with-resources)?

A Never

B When the program code doesn't terminate on its own

✓ C When there are no catch blocks in a try statement

D When there is exactly one catch block in a try statement

E When there are two or more catch blocks in a try statement

i A try statement is required to have a catch clause and/or finally clause. If it goes the catch route, it is

allowed to have multiple catch clauses.

4. Which exception will the following throw?
Object obj = new Integer(3);
String str = (String) obj;
System.out.println(str);

- A ArrayIndexOutOfBoundsException
- ✓ B ClassCastException
- C IllegalArgumentException
- D NumberFormatException
- E None of the above

i The second line tries to cast an Integer to a String. Since String does not extend Integer, this is not allowed and a ClassCastException is thrown.

5. Which of the following exceptions are thrown by the JVM? (Choose all that apply)

- ✓ A ArrayIndexOutOfBoundsException
- ✓ B ExceptionInInitializerError
- C java.io.IOException
- ✓ D NullPointerException
- E NumberFormatException

i java.io.IOException is thrown by many methods in the java.io package, but it is always thrown programmatically. The same is true for NumberFormatException; it is thrown programmatically by the wrapper classes of java.lang. The other three exceptions are all thrown by the JVM when the corresponding problem arises.

6. What will happen if you add the statement System.out.println(5 / 0); to a working main() method?

- A It will not compile.
- B It will not run.
- ✓ C It will run and throw an ArithmeticException.
- D It will run and throw an IllegalArgumentException.
- E None of the above

i The compiler tests the operation for a valid type but not a valid result, so the code will still compile and run. At runtime, evaluation of the parameter takes place before passing it to the print() method, so an ArithmeticException object is raised.

7. What is printed besides the stack trace caused by the NullPointerException from line 16?

```
1: public class DoSomething {
2: public void go() {
3: System.out.print("A");
4: try {
5: stop();
6: } catch (ArithmeticException e) {
7: System.out.print("B");
8: } finally {
9: System.out.print("C");
10: }
11: System.out.print("D");
12: }
13: public void stop() {
14: System.out.print("E");
15: Object x = null;
16: x.toString();
17: System.out.print("F");
18: }
19: public static void main(String[] args) {
20: new DoSomething().go();
21: }
22: }
```

A AE

B AEBCD

✓ **C** AEC

D AECD

E No output appears other than the stack trace.

i The main() method invokes go and A is printed on line 3. The stop method is invoked and E is printed on line 14. Line 16 throws a NullPointerException, so stop immediately ends and line 17 doesn't execute. The exception isn't caught in go, so the go method ends as well, but not before its finally block executes and C is printed on line 9. Because main() doesn't catch the exception, the stack trace displays and no further output occurs, so AEC was the output printed before the stack trace.

8. What is the output of the following code, assuming a and b are both 0?

```
3: try {
4: return a / b;
5: } catch (RuntimeException e) {
6: return -1;
7: }catch(ArithmeticException e) {
8: return 0;
9: } finally {
10: System.out.print("done");
11: }
```

A -1

B 0

C done-1

D done0

✓ E The code does not compile.

F An uncaught exception is thrown.

i The order of catch blocks is important because they're checked in the order they appear after the try block. Because ArithmeticException is a child class of RuntimeException, the catch block on line 7 is unreachable. (If an ArithmeticException is thrown in try try block, it will be caught on line 5.) Line 7 generates a compiler error because it is unreachable code.

9. What is the output of the following program?

```
1: public class Laptop {
2: public void start() {
3: try {
4: System.out.print("Starting up ");
5: throw new Exception();
6: } catch (Exception e) {
7: System.out.print("Problem ");
8: System.exit(0);
9: } finally {
10: System.out.print("Shutting down ");
11: }
12: }
13: public static void main(String[] args) {
14: new Laptop().start();
15: } }
```

A Starting up

✓ B Starting up Problem

C Starting up Problem Shutting down

D Starting up Shutting down

E The code does not compile.

F An uncaught exception is thrown.

i

The main() method invokes start on a new Laptop object. Line 4 prints Starting up; then line 5 throws an Exception. Line 6 catches the exception, line 7 prints Problem, and then line 8 calls System.exit, which terminates the JVM. The finally block does not execute because the JVM is no longer running.

10. What is the output of the following program?

```
1: public class Dog {
2: public String name;
3: public void parseName() {
4: System.out.print("1");
5: try {
6: System.out.print("2");
7: int x = Integer.parseInt(name);
8: System.out.print("3");
9: } catch (NumberFormatException e) {
10: System.out.print("4");
11: }
12: }
13: public static void main(String[] args) {
14: Dog leroy = new Dog();
15: leroy.name = "Leroy";
16: leroy.parseName();
17: System.out.print("5");
18: } }
```

A 12

B 1234

C 1235

D 124

✓ E 1245

F The code does not compile.

G An uncaught exception is thrown.

i The parseName method is invoked within main() on a new Dog object. Line 4 prints 1. The try block executes and 2 is printed. Line 7 throws a NumberFormatException, so line 8 doesn't execute. The exception is caught on line 9, and line 10 prints 4. Because the exception is handled, execution resumes normally. parseName runs to completion, and line 17 executes, printing 5. That's the end of the program, so the output is 1245.

11. What is the output of the following program?

```
1: public class Cat {
2: public String name;
3: public void parseName() {
4: System.out.print("1");
5: try {
6: System.out.print("2");
7: int x = Integer.parseInt(name);
8: System.out.print("3");
9: } catch (NullPointerException e) {
10: System.out.print("4");
11: }
12: System.out.print("5");
13: }
14: public static void main(String[] args) {
15: Cat leo = new Cat();
16: leo.name = "Leo";
17: leo.parseName();
18: System.out.print("6");
19: }
20: }
```

✓ **A** 12, followed by a stack trace for a NumberFormatException

B 124, followed by a stack trace for a NumberFormatException

C 12456

D 12456

E 1256, followed by a stack trace for a NumberFormatException

F The code does not compile.

G An uncaught exception is thrown.

i The parseName method is invoked on a new Cat object. Line 4 prints 1. The try block is entered, and line 6 prints 2. Line 7 throws a NumberFormatException. It isn't caught, so parseName ends. main() doesn't catch the exception either, so the program terminates and the stack trace for the NumberFormatException is printed.

12. What is printed by the following? (Choose all that apply)

```
1: public class Mouse {
2: public String name;
3: public void run() {
4: System.out.print("1");
5: try {
6: System.out.print("2");
7: name.toString();
8: System.out.print("3");
9: } catch (NullPointerException e) {
10: System.out.print("4");
11: throw e;
12: }
13: System.out.print("5");
14: }
15: public static void main(String[] args) {
16: Mouse jerry = new Mouse();
17: jerry.run();
18: System.out.print("6");
19: } }
```

- ✓ **A** 1
- ✓ **B** 2
- **C** 3
- ✓ **D** 4
- **E** 5
- **F** 6
- ✓ **G** The stack trace for a NullPointerException

i The main() method invokes run on a new Mouse object. Line 4 prints 1 and line 6 prints 2, so options A and B are correct. Line 7 throws a NullPointerException, which causes line 8 to be skipped, so C is incorrect. The exception is caught on line 9 and line 10 prints 4, so option D is correct. Line 11 throws the exception again, which causes run() to immediately end, so line 13 doesn't execute and option E is incorrect. The main() method doesn't catch the exception either, so line 18 doesn't execute and option F is incorrect. The uncaught NullPointerException causes the stack trace to be printed, so option G is correct.

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

- ✓ **A** You can declare a method with Exception as the return type.
- ✓ **B** You can declare any subclass of Error in the throws part of a method declaration.
- ✓ **C** You can declare any subclass of Exception in the throws part of a method declaration.
- **D** You can declare any subclass of Object in the throws part of a method declaration.
- ✓ **E** You can declare any subclass of RuntimeException in the throws part of a method declaration.

i Classes listed in the throws part of a method declaration must extend java.lang.Throwable. This includes Error, Exception, and RuntimeException. Arbitrary classes such as String can't go there. Any Java type,

including Exception, can be declared as the return type. However, this will simply return the object rather than throw an exception.

14. Which of the following can be inserted on line 8 to make this code compile? (Choose all that apply)

```
7: public void ohNo() throws IOException {
8: // INSERT CODE HERE
9: }
```

- ✓ **A** System.*out*.println("it's ok");
- **B** throw new Exception();
- ✓ **C** throw new IllegalArgumentException();
- ✓ **D** throw new java.io.IOException();
- ✓ **E** throw new RuntimeException();

i A method that declares an exception isn't required to throw one, making option A correct. Runtime exceptions can be thrown in any method, making options C and E correct. Option D matches the exception time declared and so is also correct. Option B is incorrect because a broader exception is not allowed.

15. Which of the following are unchecked exceptions? (Choose all that apply)

- ✓ **A** ArrayIndexOutOfBoundsException
- ✓ **B** IllegalArgumentException
- **C** IOException
- ✓ **D** NumberFormatException
- ✓ **E** Any exception that extends RuntimeException
- **F** Any exception that extends Exception

i ArrayIndexOutOfBoundsException, IllegalArgumentException, and NumberFormatException are runtime exceptions. Sorry, you have to memorize them. Any class that extends RuntimeException is a runtime (unchecked) exception. Classes that extend Exception but not RuntimeException are checked exceptions.

16. Which scenario is the best use of an exception?

- **A** An element is not found when searching a list.
- ✓ **B** An unexpected parameter is passed into a method.
- **C** The computer caught fire.
- **D** You want to loop through a list.
- **E** You don't know how to code a method.

i IllegalArgumentException is used when an unexpected parameter is passed into a method. Option A is incorrect because returning null or -1 is a common return value for this scenario. Option D is incorrect because a for loop is typically used for this scenario. Option E is incorrect because you should find out how to code the method and not leave it for the unsuspecting programmer who calls your method. Option C is incorrect because you should run!

17. Which of the following can be inserted into Lion to make this code compile? (Choose all that apply)

```
class HasSoreThroatException extends Exception {}
class TiredException extends RuntimeException {}
interface Roar {
void roar() throws HasSoreThroatException;
}
class Lion implements Roar {// INSERT CODE HERE
}
```

- ✓ **A** public void roar(){}
- **B** public void roar() throws Exception{}
- ✓ **C** public void roar() throws HasSoreThroatException{}
- ✓ **D** public void roar() throws IllegalArgumentException{}
- ✓ **E** public void roar() throws TiredException{}

> **i** The method is allowed to throw no exceptions at all, making option A correct. It is also allowed to throw runtime exceptions, making options D and E correct. Option C is also correct since it matches the signature in the interface.

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

- **A** Checked exceptions are allowed to be handled or declared.
- ✓ **B** Checked exceptions are required to be handled or declared.
- ✓ **C** Errors are allowed to be handled or declared.
- **D** Errors are required to be handled or declared.
- ✓ **E** Runtime exceptions are allowed to be handled or declared.
- **F** Runtime exceptions are required to be handled or declared.

> **i** Checked exceptions are required to be handled or declared. Runtime exceptions are allowed to be handled or declared. Errors are allowed to be handled or declared, but this is bad practice.

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

```
public static void main(String[] args) {
try {
System.out.println("work real hard");
} catch (_____ e) {
} catch (RuntimeException e) {
}
}
```

- A Exception
- B IOException
- ✓ C IllegalArgumentException
- D RuntimeException
- ✓ E StackOverflowError
- F None of the above

i Option C is allowed because it is a more specific type than RuntimeException. Option E is allowed because it isn't in the same inheritance tree as RuntimeException. It's not a good idea to catch either of these. Option B is not allowed because the method called inside the try block doesn't declare an IOException to be thrown. The compiler realizes that IOException would be an unreachable catch block. Option D is not allowed because the same exception can't be specified in two different catch blocks. Finally, option A is not allowed because it's more general than RuntimeException and would make that block unreachable.

20. What does the output of the following contain? (Choose all that apply)

```
12: public static void main(String[] args) {
13: System.out.print("a");
14: try {
15: System.out.print("b");
16: throw new IllegalArgumentException();
17: } catch (IllegalArgumentException e) {
18: System.out.print("c");
19: throw new RuntimeException("1");
20: } catch (RuntimeException e) {
21: System.out.print("d");
22: throw new RuntimeException("2");
23: } finally {
24: System.out.print("e");
25: throw new RuntimeException("3");
26: }
27: }
```

- ✓ **A** abce
- **B** abde
- **C** An exception with the message set to "1"
- **D** An exception with the message set to "2"
- ✓ **E** An exception with the message set to "3"
- **F** Nothing; the code does not compile.

i The code begins normally and prints a on line 13, followed by b on line 15. On line 16, it throws an exception that's caught on line 17. Remember only the most specific matching catch is run. Line 18 prints c, and then line 19 throws another exception. Regardless, the finally block runs, printing e. Since the finally block also throws an exception, that's the one printed.