Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

Section 12.2

▼12.2.1

What is the advantage of using exception handling?

The advantages of using exception handling: It enables a method to throw an exception to its caller. The caller can handle this exception. Without this capability, the called method itself must handle the exception or terminate the program. Often the called method does not know how to handle the exception. So it needs to pass the exception to its caller for handling.

Hide Answer

▼12.2.2

Which of the following statements will throw an exception?

```
System.out.println(1 / 0);
System.out.println(1.0 / 0);
System.out.println(1 / 0); // Throws an exception
System.out.println(1.0 / 0); // Will not throw an exception
```

Hide Answer

▼12.2.3

Point out the problem in the following code. Does the code throw any exceptions?

```
long value = Long.MAX_VALUE + 1;
System.out.println(value);
```

Adding 1 to Long.MAX_VALUE exceeds the maximum value allowed by a long value. But the current versions of Java does not report this as an exception.

Hide Answer

▼12.2.4

What does the JVM do when an exception occurs? How do you catch an exception?

When an excepion occurs, Java searches for a handler in the catch clause. So to catch an exception in your program, you need to write a try-catch statement like this:

```
try {

}
catch (Exception ex) {
  // Catch and process exception
}
```

Hide Answer

▼12.2.5

What is the output of the following code?

```
public class Test {
  public static void main(String[] args) {
    try {
      int value = 30;
      if (value < 40)
         throw new Exception("value is too small");
    }
    catch (Exception ex) {
      System.out.println(ex.getMessage());
    }
    System.out.println("Continue after the catch block");
  }
}
```

What would be the output if the line

```
int value = 30;
```

were changed to

```
int value = 50;
```

output is

```
value is too small
Continue after the catch block
```

The output would be if Line int value = 30; is changed to int value = 50; Continue after the catch block

Hide Answer

▼12.2.6

Show the output of the following code.

(a)

```
public class Test {
  public static void main(String[] args) {
    for (int i = 0; i < 2; i++) {
      System.out.print(i + " ");
      try {
        System.out.println(1 / 0);
      }
      catch (Exception ex) {
      }
    }
  }
}
```

(b)

```
public class Test {
  public static void main(String[] args) {
    try {
      for (int i = 0; i < 2; i++) {
        System.out.print(i + " ");
```

```
        System.out.println(1 / 0);
      }
    }
    catch (Exception ex) {
    }
  }
}
```

a. 0 1
b. 0

Hide Answer

Section 12.3

▼12.3.1

Describe the Java Throwable class, its subclasses, and the types of exceptions.

See the section "Exceptions and Exception Types." The Throwable class is the root of Java exception classes. Error and Exception are subclasses of Throwable. Error describes fatal system errors, and Exception describes the errors that can be handled by Java programs. The subclasses of Error are LinkageError, VirtualMachineError, and AWTError. The subclasses of Exception include RuntimeException, IOException, AWTException, and InstantiationException.

Hide Answer

▼12.3.2

What RuntimeException will the following programs throw, if any?

(a)
```
public class Test {
  public static void main(String[] args) {
    System.out.println(1 / 0);
  }
}
```

(b)
```
public class Test {
  public static void main(String[] args) {
    int[] list = new int[5];
    System.out.println(list[5]);
  }
}
```

(c)
```
public class Test {
  public static void main(String[] args) {
    String s = "abc";
    System.out.println(s.charAt(3));
  }
}
```

(d)
```
public class Test {
```

```
  public static void main(String[] args) {
    Object o = new Object();
    String d = (String)o;
  }
}
```

```
(e)
public class Test {
  public static void main(String[] args) {
    Object o = null;
    System.out.println(o.toString());
  }
}
```

```
(f)
public class Test {
  public static void main(String[] args) {
    System.out.println(1.0 / 0);
  }
}
```

a. ArithmeticException
b. ArrayIndexOutOfBoundsException
c. StringIndexOutOfBoundsException
d. ClassCastException
e. NullPointerException
f. No exception

Hide Answer

Section 12.4

▼12.4.1

What is the purpose of declaring exceptions? How do you declare an exception, and where? Can you declare multiple exceptions in a method header?

The purpose of declaring exceptions is to tell the Java runtime system what can go wrong. You declare an exception using the throws keyword in the method declaration. You can declare multiple exceptions, separated by commas.

Hide Answer

▼12.4.2

What is a checked exception, and what is an unchecked exception?

A checked exception must be explicitly declared in the method declaration, if a method throws it. A checked exception must be caught in a try-catch block. An unchecked exception does not need to be declared and does not need to be caught. In Java, the only unchecked exceptions are RuntimeException and Error and their subclasses.

Hide Answer

▼12.4.3

How do you throw an exception? Can you throw multiple exceptions in one throw statement?

You use the throw statement in the method to throw an exception. You cannot throw multiple exceptions in a single throw statement.

Hide Answer

▼12.4.4

What is the keyword throw used for? What is the keyword throws used for?

throw is for throwing exceptions and throws is for claiming exceptions.

Hide Answer

▼12.4.5

Suppose that statement2 causes an exception in the following try-catch block:

```
try {
  statement1;
  statement2;
  statement3;
}
catch (Exception1 ex1) {
}
catch (Exception2 ex2) {
}

statement4;
```

Answer the following questions:

1. Will statement3 be executed?
2. If the exception is not caught, will statement4 be executed?
3. If the exception is caught in the catch block, will statement4 be executed?

1. No.
2. No.
3. Yes.

Hide Answer

▼12.4.6

What is displayed when running the following program?

```
public class Test {
  public static void main(String[] args) {
    try {
      int[] list = new int[10];
      System.out.println("list[10] is " + list[10]);
    }
    catch (ArithmeticException ex) {
      System.out.println("ArithmeticException");
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
    }
    catch (Exception ex) {
```

```
      System.out.println("Exception");
    }
  }
}
```

RuntimeException

Reason: list[10] throws ArrayIndexOutOfBoundsException that is a subclass of RuntimeException.

Hide Answer

▼12.4.7

What is displayed when running the following program?

```
public class Test {
  public static void main(String[] args) {
    try {
      method();
      System.out.println("After the method call");
    }
    catch (ArithmeticException ex) {
      System.out.println("ArithmeticException");
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException");
    }
    catch (Exception e) {
      System.out.println("Exception");
    }
  }

  static void method() throws Exception {
    System.out.println(1 / 0);
  }
}
```

ArithmeticException

Reason: method() throws ArithmeticException.

Hide Answer

▼12.4.8

What is displayed when running the following program?

```
public class Test {
  public static void main(String[] args) {
    try {
      method();
      System.out.println("After the method call");
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException in main");
    }
    catch (Exception ex) {
      System.out.println("Exception in main");
    }
```

```
  }

  static void method() throws Exception {
    try {
      String s ="abc";
      System.out.println(s.charAt(3));
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException in method()");
    }
    catch (Exception ex) {
      System.out.println("Exception in method()");
    }
  }
}
```

```
RuntimeException in method()
After the method call
```

Reason: s.charAt(3) throws StringIndexOutOfBoundsException that is a subclass of RuntimeException. This exception is caught in method(). The main method continues its normal execution flow.

Hide Answer

▼12.4.9

What does the method getMessage() do?

the getMessage() is defined in the Throwable class to return a string that describes the exception.

Hide Answer

▼12.4.10

What does the method printStackTrace() do?

To display trace information to the console.

Hide Answer

▼12.4.11

Does the presence of a try-catch block impose overhead when no exception occurs?

No

Hide Answer

▼12.4.12

Correct a compile error in the following code:

```
public void m(int value) {
  if (value < 40)
    throw new Exception("value is too small");
}
```

The method throws a checked exception. It must be caught or thrown. You may fix it as follows:

```
public void m(int value) throws Exception {
  if (value < 40)
```

```
    throw new Exception("value is too small");
}
```

Hide Answer

Section 12.5

▼12.5.1

Suppose you run the following code:

```
public static void main(String[] args) throws Exception2 {
  m();
  statement7;
}

public static void m() {
  try {
    statement1;
    statement2;
    statement3;
  }
  catch (Exception1 ex1) {
    statement4;
  }
  finally {
    statement5;
  }
  statement6;
}
```

answer the questions:

1. If no exception occurs, which statements are executed?
2. If statement2 throws an exception of type Exception1, which statements are executed?
3. If statement2 throws an exception of type Exception2, which statements are executed?
4. If statement2 throws an exception that is neither Exception1 nor Exception2, which statements are executed?

1. statement1, statement2, statement3, statement5, statement6, statement7.
2. statement1, statement2, statement4, statement5, statement6, statement7.
3. statement1, statement2, statement5.
4. statement1, statement2, statement5.

Hide Answer

▼12.5.2

Suppose you run the following code:

```
public static void main(String[] args) {
  try {
    m();
    statement7;
  }
```

```
  catch (Exception2 ex {
    statement8;
  }
}

public static void m() {
  try {
    statement1;
    statement2;
    statement3;
  }
  catch (Exception1 ex1) {
    statement4;
  }
  finally {
    statement5;
  }
  statement6;
}
```

answer the questions:

1. If no exception occurs, which statements are executed?
2. If statement2 throws an exception of type Exception1, which statements are executed?
3. If statement2 throws an exception of type Exception2, which statements are executed?
4. If statement2 throws an exception that is neither Exception1 nor Exception 2, which statements are executed?

1. statement1, statement2, statement3, statement5, statement6, statement7.
2. statement1, statement2, statement4, statement5, statement6, statement7.
3. statement1, statement2, statement5, statement8.
4. statement1, statement2, statement5.

Hide Answer

Section 12.6

▼12.6.1

The following method tests whether a string is a numeric string:

```
public static boolean isNumeric(String token) {
  try {
    Double.parseDouble(token);
    return true;
  }
  catch (java.lang.NumberFormatException ex) {
    return false;
  }
}
```

Is it correct? Rewrite it without using exceptions.

Yes. It is correct. But a better way to write this method is using regular expression to test if the string is numeric.

Hide Answer

Section 12.7

▼12.7.1

Suppose that statement2 may cause an exception in the following code:

```
try {
  statement1;
  statement2;
  statement3;
}
catch (Exception1 ex1) {
}
catch (Exception2 ex2) {
  throw ex2;
}
finally {
  statement4;
}
statement5;
```

Answer the following questions:

1. If no exception occurs, will statement4 be executed, and will statement5 be executed?
2. If the exception is of type Exception1, will statement4 be executed, and will statement5 be executed?
3. If the exception is of type Exception2, will statement4 be executed, and will statement5 be executed?
4. If the exception is not Exception1 nor Exception2, will statement4 be executed, and will statement5 be executed?

1. Yes to both.
2. Yes to both.
3. This exception is caught by the catch (Exception2 e2) clause and statement4 will be executed, but statement5 will not be executed because it is rethrown to its caller.
4. This exception is not caught. statement4 will be executed, but statement5 will not be executed.

Hide Answer

Section 12.8

▼12.8.1

What would be the output if line 16 is replaced by the following line?

```
throw new Exception("New info from method1");
```

The output will be

```
java.lang.Exception: New info from method1
at ChainedExceptionDemo.method1(ChainedExceptionDemo.java:16)
at ChainedExceptionDemo.main(ChainedExceptionDemo.java:4)
```

Hide Answer

Section 12.9

▼12.9.1

How do you define a custom exception class?

To define a custom class, extend Exception or a subclass of Exception.

Hide Answer

▼12.9.2

Suppose the setRadius method throws the InvalidRadiusException defined in Listing 12.10. What is displayed when running the following program?

```
public class Test {
  public static void main(String[] args) {
    try {
      method();
      System.out.println("After the method call");
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException in main");
    }
    catch (Exception ex) {
      System.out.println("Exception in main");
    }
  }

  static void method() throws Exception {
    try {
      Circle c1 = new Circle(1);
      c1.setRadius(-1);
      System.out.println(c1.getRadius());
    }
    catch (RuntimeException ex) {
      System.out.println("RuntimeException in method()");
    }
    catch (Exception ex) {
      System.out.println("Exception in method()");
      throw ex;
    }
  }
}
```

```
Exception in method()
Exception in main
```

Reason: the setRadius method throws a RadiusException. RadiusException is a subclass of Exception. So it is caught in method()'s handler. The handler rethrows it back to the main method.

Hide Answer

Section 12.10

▼12.10.1

What is wrong about creating a File object using the following statement?

```
new File("c:\book\test.dat");
```

The \ is a special character. It should be written as \\ in Java using the Escape sequence.

Hide Answer

▼12.10.2

How do you check whether a file already exists? How do you delete a file? How do you rename a file? Can you find the file size (the number of bytes) using the File class? How do you create a directory?

Use exists() in the File class to check whether a file exists. Use delete() in the File class to delete this file. Use renameTo(File) to rename the name for this file. Use length() to return the file size. Use mkdir or mkdirs to create a directory under this File object.

Hide Answer

▼12.10.3

Can you use the File class for I/O? Does creating a File object create a file on the disk?

No. The File class can be used to obtain file properties and manipulate files, but cannot perform I/O. No. Creating a File object does not create a file/directory on the disk.

Hide Answer

Section 12.11

▼12.11.1

How do you create a PrintWriter to write data to a file? What is the reason to declare throws Exception in the main method in Listing 12.13, WriteData.java? What would happen if the close() method were not invoked in Listing 12.13?

To create a PrintWriter for a file, use new PrintWriter(filename). This statement may throw an exception. Java forces you to write the code to deal with exceptions. One way to deal with it is to declare throws java.io.IOException in the method declaration. If the close() method is not invoked, the data may not be saved properly.

Hide Answer

▼12.11.2

Show the contents of the file temp.txt after the following program is executed.

```
public class Test {
  public static void main(String[] args) throws java.io.IOException {
    java.io.PrintWriter output = new java.io.PrintWriter("temp.txt");
    output.printf("amount is %f %e\r\n", 32.32, 32.32);
    output.printf("amount is %5.4f %5.4e\r\n", 32.32, 32.32);
    output.printf("%6b\r\n", (1 > 2));
    output.printf("%6s\r\n", "Java");
    output.close();
```

```
  }
}
```

The contents of the file temp.txt is:

```
amount is 32.320000 3.232000e+01
amount is 32.3200 3.2320e+01
 false
  Java
```

Hide Answer

▼12.11.3

Rewrite the code in the preceding question using a try-with-resources syntax.

```
public class Test {
  public static void main(String[] args) throws Exception {
    try (java.io.PrintWriter output = new
      java.io.PrintWriter("temp.txt");) {
      output.printf("amount is %f %e\r\n", 32.32, 32.32);
      output.printf("amount is %5.4f %5.4e\r\n", 32.32, 32.32);
      output.printf("%6b\r\n", (1 > 2));
      output.printf("%6s\r\n", "Java");
    }
  }
}
```

Hide Answer

▼12.11.4

How do you create a Scanner to read data from a file? What is the reason to define throws Exception in the main method in Listing 12.15, ReadData.java? What would happen if the close() method were not invoked in Listing 12.15?

To create a Scanner for a file, use new Scanner(new File(filename)). This statement may throw an exception. Java forces you to write the code to deal with exceptions. One way to deal with it is to declare throws Exception in the method declaration. If the close() method is not invoked, the problem will run fine. But it is a good practice to close the file to release the resource on the file.

Hide Answer

▼12.11.5

What will happen if you attempt to create a Scanner for a nonexistent file? What will happen if you attempt to create a PrintWriter for an existing file?

If you attempt to create a Scanner for a nonexistent file, an exception will occur. If you attempt to create a PrintWriter for an existing file, the contents of the existing file will be gone.

Hide Answer

▼12.11.6

Is the line separator the same on all platforms? What is the line separator on Windows?

No. The line separator on Windows is \r\n.

Hide Answer

▼12.11.7

Suppose you enter 45 57.8 789, then press the Enter key. Show the contents of the variables after the following code is executed.

```
Scanner input = new Scanner(System.in);
int intValue = input.nextInt();
double doubleValue = input.nextDouble();
String line = input.nextLine();
```

intValue contains 45. doubleValue contains 57.8, and line contains ' ', '7 ', '8 ', '9'.

Hide Answer

▼12.11.8

Suppose you enter 45, press the Enter key, 57.8, press the Enter key, 789, and press the Enter key. Show the contents of the variables after the following code is executed.

```
Scanner input = new Scanner(System.in);
int intValue = input.nextInt();
double doubleValue = input.nextDouble();
String line = input.nextLine();
```

intValue contains 45. doubleValue contains 57.8, and line is empty.

Hide Answer

Section 12.12

▼12.12.1

How do you create a Scanner object for reading text from a URL?

Create a URL object and use new Scanner(url.openStream()) to create a scanner object for reading data from the URL stream.

Hide Answer

Section 12.13

▼12.13.1

Before a URL is added to listOfPendingURLs, line 25 tests whether it has been traversed. Is it possible that listOfPendingURLs contains duplicate URLs? If so, give an example.

Yes. Possible. Suppose link1 is not in listOfTraversedURLs, but it appears more than one time in a page. Duplicate link1 will be added into listOfPendingURLs.

Hide Answer

▼12.13.2

Simplify the code in lines 20-28 as follows: 1. Delete lines 20 and 28; 2. Add an additional condition !listOfPendingURLs.contains(s) to the if statement in line 25. Write the complete new code for the while loop in lines 17-29. Does this revision work?

The complete new code is

```
    while (!listOfPendingURLs.isEmpty() &&
        listOfTraversedURLs.size() <= 100) {
```

```
      String urlString = listOfPendingURLs.remove(0);
      listOfTraversedURLs.add(urlString);
      System.out.println("Crawl " + urlString);

      for (String s: getSubURLs(urlString)) {
        if (!listOfTraversedURLs.contains(s) &&
            !listOfPendingURLs.contains(s))
          listOfPendingURLs.add(s);
      }
    }
```

The revison works and simplifies the code. It ensures that no duplicate URL strings are added to listOfPendingURLs.

Hide Answer