Exception Handling Exercises and self-Evaluation

1. Division by Zero?

import java.util.Scanner;

public class Quotient {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Prompt the user to enter two integers

System.out.print("Enter two integers: ");

int number1 = input.nextInt();

int number2 = input.nextInt();

System.out.println(number1 + " / " + number2 + " is " +

(number1 / number2));

}

}

1. Handling exception with a if

import java.util.Scanner;

public class QuotientWithIf {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Prompt the user to enter two integers

System.out.print("Enter two integers: ");

int number1 = input.nextInt();

int number2 = input.nextInt();

if (number2 != 0)

System.out.println(number1 + " / " + number2 + " is " +

(number1 / number2));

else

System.out.println("Divisor cannot be zero ");

}

}

1. An erroneous situation and a method

import java.util.Scanner;

public class QuotientWithMethod {

public static int quotient(int number1, int number2) {

if (number2 == 0)

throw new ArithmeticException("Divisor cannot be zero");

return number1 / number2;

}

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

// Prompt the user to enter two integers

System.out.print("Enter two integers: ");

int number1 = input.nextInt();

int number2 = input.nextInt();

try {

int result = quotient(number1, number2);

System.out.println(number1 + " / " + number2 + " is "

+ result);

}

catch (Exception ex) {

System.out.println("Exception: an integer " +

"cannot be divided by zero ");

}

System.out.println("Execution continues ...");

}

}

1. Handling Input mismatches

import java.util.\*;

public class InputMismatchExceptionDemo {

public static void main(String[] args) {

Scanner input = new Scanner(System.in);

boolean continueInput = true;

do {

try {

System.out.print("Enter an integer: ");

int number = input.nextInt();

// Display the result

System.out.println(

"The number entered is " + number);

continueInput = false;

}

catch (InputMismatchException ex) {

System.out.println("Try again. (" +

"Incorrect input: an integer is required)");

input.nextLine(); // discard input

}

} while (continueInput);

}

}

1. Example: Declaring, Throwing, and Catching Exceptions
   1. circle with exceptions

public class CircleWithException {

/\*\* The radius of the circle \*/

private double radius;

/\*\* The number of the objects created \*/

private static int numberOfObjects = 0;

/\*\* Construct a circle with radius 1 \*/

public CircleWithException() {

this(1.0);

}

/\*\* Construct a circle with a specified radius \*/

public CircleWithException(double newRadius) {

setRadius(newRadius);

numberOfObjects++;

}

/\*\* Return radius \*/

public double getRadius() {

return radius;

}

/\*\* Set a new radius \*/

public void setRadius(double newRadius)

throws IllegalArgumentException {

if (newRadius >= 0)

radius = newRadius;

else

throw new IllegalArgumentException(

"Radius cannot be negative");

}

/\*\* Return numberOfObjects \*/

public static int getNumberOfObjects() {

return numberOfObjects;

}

/\*\* Return the area of this circle \*/

public double findArea() {

return radius \* radius \* 3.14159;

}

}

* 1. Test Circle with exceptions

public class TestCircleWithException {

public static void main(String[] args) {

try {

CircleWithException c1 = new CircleWithException(5);

CircleWithException c2 = new CircleWithException(-5);

CircleWithException c3 = new CircleWithException(0);

}

catch (IllegalArgumentException ex) {

System.out.println(ex);

}

System.out.println("Number of objects created: " +

CircleWithException.getNumberOfObjects());

}

}

1. Invalid radius demo

public class InvalidRadiusException extends Exception {

private double radius;

/\*\* Construct an exception \*/

public InvalidRadiusException(double radius) {

super("Invalid radius " + radius);

this.radius = radius;

}

/\*\* Return the radius \*/

public double getRadius() {

return radius;

}

}



|  |  |  |
| --- | --- | --- |
| Consider the following statements on the exceptions class hierarchy and mark (the) incorrect statement/s. | | |
|  | 1. Instances of Error class occur in a Java program because of code that is not very robust. |
|  | 1. Run Time Exceptions are internal errors in the Java run time environment. |
|  | 1. Two sub classes of RunTimeExceptions class are EOFException and FileNotFoundExceptions |
|  | 1. Exceptions of either Error or RunTimeException classes do not have to be listed in the throws clause and they are called implicit exceptions. |
|  | 1. IOExceptions are part of java.lang package. |



|  |  |  |
| --- | --- | --- |
| In Java programming language errors and other unusual conditions are managed by: | | |
| 1. using String class. | 1. using different control structures. |
| 1. using special language features in Java. | 1. consistency checking in compile time. |
| 1. by using a set of extensible exception classes. |  |



|  |  |
| --- | --- |
| Select correct statement/s from among the following on Exceptions. | |
|  | |
|  | 1. Exceptions are all instances of some exception class. |
|  | 1. In Java one can create one’s own exception to deal with a problematic situation. |
|  | 1. When creating a new exception it should inherit from some other exception in the Java hierarchy. |
|  | 1. All user created exceptions should be a part of Error hierarchy. |
|  | 1. Exception classes do not have constructors. |

10. When does Exceptions in Java arises in code sequence?  
a) Run Time  
b) Compilation Time  
c) Can Occur Any Time  
d) None of the mentioned  
View Answer

Answer: a  
Explanation: Exceptions in java are run-time errors.

11. Which of these keywords is not a part of exception handling?  
a) try  
b) finally  
c) thrown  
d) catch  
View Answer

Answer: c  
Explanation: Exceptional handling is managed via 5 keywords – try, catch, throws, throw and finally.

12. Which of these keywords must be used to monitor for exceptions?  
a) try  
b) finally  
c) throw  
d) catch  
View Answer

Answer: a  
Explanation: None.

13. Which of these keywords must be used to handle the exception thrown by try block in some rational manner?  
a) try  
b) finally  
c) throw  
d) catch  
View Answer

Answer: d  
Explanation: If an exception occurs within the try block, it is thrown and cached by catch block for processing.

14. Which of these keywords is used to manually throw an exception?  
a) try  
b) finally  
c) throw  
d) catch  
View Answer

Answer: c  
Explanation: None.

15. What is the output of this program?

1. class exception\_handling {
2. public static void main(String args[]) {
3. try {
4. System.out.print("Hello" + " " + 1 / 0);
5. }
6. catch(ArithmeticException e) {
7. System.out.print("World");
8. }
9. }
10. }

a) Hello  
b) World  
c) HelloWorld  
d) Hello World  
View Answer

Answer: b  
Explanation: System.ou.print() function fist converts the whole parameters into string and then prints, before “Hello” goes to output stream 1 / 0 error is encountered which is cached by catch block printing just “World” .  
Output:  
$ javac exception\_handling.java  
$ java exception\_handling  
World

16. What is the output of this program?

1. class exception\_handling {
2. public static void main(String args[]) {
3. try {
4. int a, b;
5. b = 0;
6. a = 5 / b;
7. System.out.print("A");
8. }
9. catch(ArithmeticException e) {
10. System.out.print("B");
11. }
12. }
13. }

a) A  
b) B  
c) Compilation Error  
d) Runtime Error  
View Answer

Answer: b  
Explanation: None.  
Output:  
$ javac exception\_handling.java  
$ java exception\_handling  
B

17. What is the output of this program?

1. class exception\_handling {
2. public static void main(String args[]) {
3. try {
4. int a, b;
5. b = 0;
6. a = 5 / b;
7. System.out.print("A");
8. }
9. catch(ArithmeticException e) {
10. System.out.print("B");
11. }
12. finally {
13. System.out.print("C");
14. }
15. }
16. }

a) A  
b) B  
c) AC  
d) BC  
View Answer

Answer: d  
Explanation: finally keyword is used to execute the code before try and catch block end.  
Output:  
$ javac exception\_handling.java  
$ java exception\_handling  
BC

18. What is the output of this program?

1. class exception\_handling {
2. public static void main(String args[]) {
3. try {
4. int i, sum;
5. sum = 10;
6. for (i = -1; i < 3 ;++i)
7. sum = (sum / i);
8. }
9. catch(ArithmeticException e) {
10. System.out.print("0");
11. }
12. System.out.print(sum);
13. }
14. }

a) 0  
b) 05  
c) Compilation Error  
d) Runtime Error  
View Answer

Answer: b  
Explanation: Value of variable sum is printed outside of try block, sum is declared only in try block, outside try block it is undefined.  
Output:  
$ javac exception\_handling.java  
Exception in thread “main” java.lang.Error: Unresolved compilation problem:  
sum cannot be resolved to a variable

19. What is the output of this program?

1. class exception\_handling {
2. public static void main(String args[]) {
3. try {
4. int i, sum;
5. sum = 10;
6. for (i = -1; i < 3 ;++i) {
7. sum = (sum / i);
8. System.out.print(i);
9. }
10. }
11. catch(ArithmeticException e) {
12. System.out.print("0");
13. }
14. }
15. }

a) -1  
b) 0  
c) -10  
d) -101  
View Answer

Answer: c  
Explanation: None.  
Output:  
$ javac exception\_handling.java  
$ java exception\_handling  
BC

Assertions Exercises and self-Evaluation

|  |  |
| --- | --- |
| 1. | What will be the output of the program?  public class Test  {  public static void main(String[] args)  {  int x = 0;  assert (x > 0) ? "assertion failed" : "assertion passed" ;  System.out.println("finished");  }  } |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | finished | | [B.](javascript:%20void%200;) | Compiliation fails. | | [C.](javascript:%20void%200;) | An *AssertionError* is thrown and finished is output. | | [D.](javascript:%20void%200;) | An *AssertionError* is thrown with the message "assertion failed." |   [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **B**  **Explanation:**  Compilation Fails. You can't use the Assert statement in a similar way to the ternary operator. Don't confuse. |

|  |  |
| --- | --- |
| 2. | public class Test  {  public void foo()  {  assert false; /\* Line 5 \*/  assert false; /\* Line 6 \*/  }  public void bar()  {  while(true)  {  assert false; /\* Line 12 \*/  }  assert false; /\* Line 14 \*/  }  }  What causes compilation to fail? |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | Line 5 | | [B.](javascript:%20void%200;) | Line 6 | | [C.](javascript:%20void%200;) | Line 12 | | [D.](javascript:%20void%200;) | Line 14 |   [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **D**  **Explanation:**  Option D is correct. Compilation fails because of an unreachable statement at line 14. It is a compile-time error if a statement cannot be executed because it is unreachable. The question is now, why is line 20 unreachable? If it is because of the assert then surely line 6 would also be unreachable. The answer must be something other than assert.  Examine the following:  A while statement can complete normally if and only if at least one of the following is true:  - The *while* statement is reachable and the condition expression is not a constant expression with value true.  -There is a reachable break statement that exits the *while* statement.  The while statement at line 11 is infinite and there is no break statement therefore line 14 is unreachable. You can test this with the following code:  public class Test80  {  public void foo()  {  assert false;  assert false;  }  public void bar()  {  while(true)  {  assert false;  break;  }  assert false;  }  } |

|  |  |
| --- | --- |
| 3. | What will be the output of the program?  public class Test  {  public static int y;  public static void foo(int x)  {  System.out.print("foo ");  y = x;  }  public static int bar(int z)  {  System.out.print("bar ");  return y = z;  }  public static void main(String [] args )  {  int t = 0;  assert t > 0 : bar(7);  assert t > 1 : foo(8); /\* Line 18 \*/  System.out.println("done ");  }  } |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | bar | | [B.](javascript:%20void%200;) | bar done | | [C.](javascript:%20void%200;) | foo done | | [D.](javascript:%20void%200;) | Compilation fails |   [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **D**  **Explanation:**  The *foo()* method returns void. It is a perfectly acceptable method, but because it returns void it cannot be used in an *assert* statement, so line 18 will not compile. |

|  |  |
| --- | --- |
| 4. | What will be the output of the program (when you run with the -ea option) ?  public class Test  {  public static void main(String[] args)  {  int x = 0;  assert (x > 0) : "assertion failed"; /\* Line 6 \*/  System.out.println("finished");  }  } |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | finished | | [B.](javascript:%20void%200;) | Compilation fails. | | [C.](javascript:%20void%200;) | An AssertionError is thrown. | | [D.](javascript:%20void%200;) | An AssertionError is thrown and finished is output. |   [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **C**  **Explanation:**  An assertion Error is thrown as normal giving the output "assertion failed". The word "finished" is not printed (ensure you run with the *-ea* option)  Assertion failures are generally labeled in the stack trace with the file and line number from which they were thrown, and also in this case with the error's detail message "assertion failed". The detail message is supplied by the assert statement in line 6. |

|  |  |
| --- | --- |
| 5. | public class Test2  {  public static int x;  public static int foo(int y)  {  return y \* 2;  }  public static void main(String [] args)  {  int z = 5;  assert z > 0; /\* Line 11 \*/  assert z > 2: foo(z); /\* Line 12 \*/  if ( z < 7 )  assert z > 4; /\* Line 14 \*/  switch (z)  {  case 4: System.out.println("4 ");  case 5: System.out.println("5 ");  default: assert z < 10;  }  if ( z < 10 )  assert z > 4: z++; /\* Line 22 \*/  System.out.println(z);  }  }  which line is an example of an inappropriate use of assertions? |
| |  |  | | --- | --- | | [A.](javascript:%20void%200;) | Line 11 | | [B.](javascript:%20void%200;) | Line 12 | | [C.](javascript:%20void%200;) | Line 14 | | [D.](javascript:%20void%200;) | Line 22 |   [Answer & Explanation](javascript:%20void%200;)  **Answer:** Option **D**  **Explanation:**  Assert statements should not cause side effects. Line 22 changes the value of *z* if the assert statement is *false*.  Option A is fine; a second expression in an assert statement is not required.  Option B is fine because it is perfectly acceptable to call a method with the second expression of an assert statement.  Option C is fine because it is proper to call an assert statement conditionally. |