OCP Java SE 8

Assertions und Exceptions

Assertions

- Zusicherungen
- Schlüsselwort assert
- Wenn Bedingung nicht erfüllt wird java.lang.AssertionError geworfen und JVM beendet.
- Einsatz nur bei Entwicklung/Debuging
- Aktivieren: -ea oder -enableassertions
- Deaktivieren: -da oder -disableassertions

Assertions

- java [-enableassertions | -ea] [:Paket... | :Klasse]
- java [-disableassertions | -da] [:Paket... | :Klassen]

Assertions

- java -ea de.gfn.AssertionTest
- java -ea -da:de.gfn.AssertionTest
- java -ea -da:de.gfn…

Assertions

- bis Java 1.3 war assert kein Schlüsselwort und konnte als Bezeichnet verwendet werden
- javac -source 1.3 MyClass.java

Assertions

- Preconditions

```
public String test() {
  assert intVar == 0;
  //...
}
```

Assertions

- Postconditions

```
public String test() {
  //...
  assert intVar == 0;
}
```

Assertions

- 2 Varianten
  - `assert Condition;`
  - `assert Condition : Message`
    - Condition muss ein boolean liefern
    - bei false in der Condition gibt es ein AssertionError. Message ist optional und wird an das Stack-Trace weitergegeben

Wichtig!

- AssertionError sollte niemals behandelt werden
- Sollte nicht zum Validieren von public Methoden genutzt werden (lieber IllegalArgumentException verwenden)
- Sollte nicht zum Validieren von Command-Line-Argumenten genutzt werden
- Kann in privaten Methoden oder in public Methoden für Fälle die nicht auftreten sollten

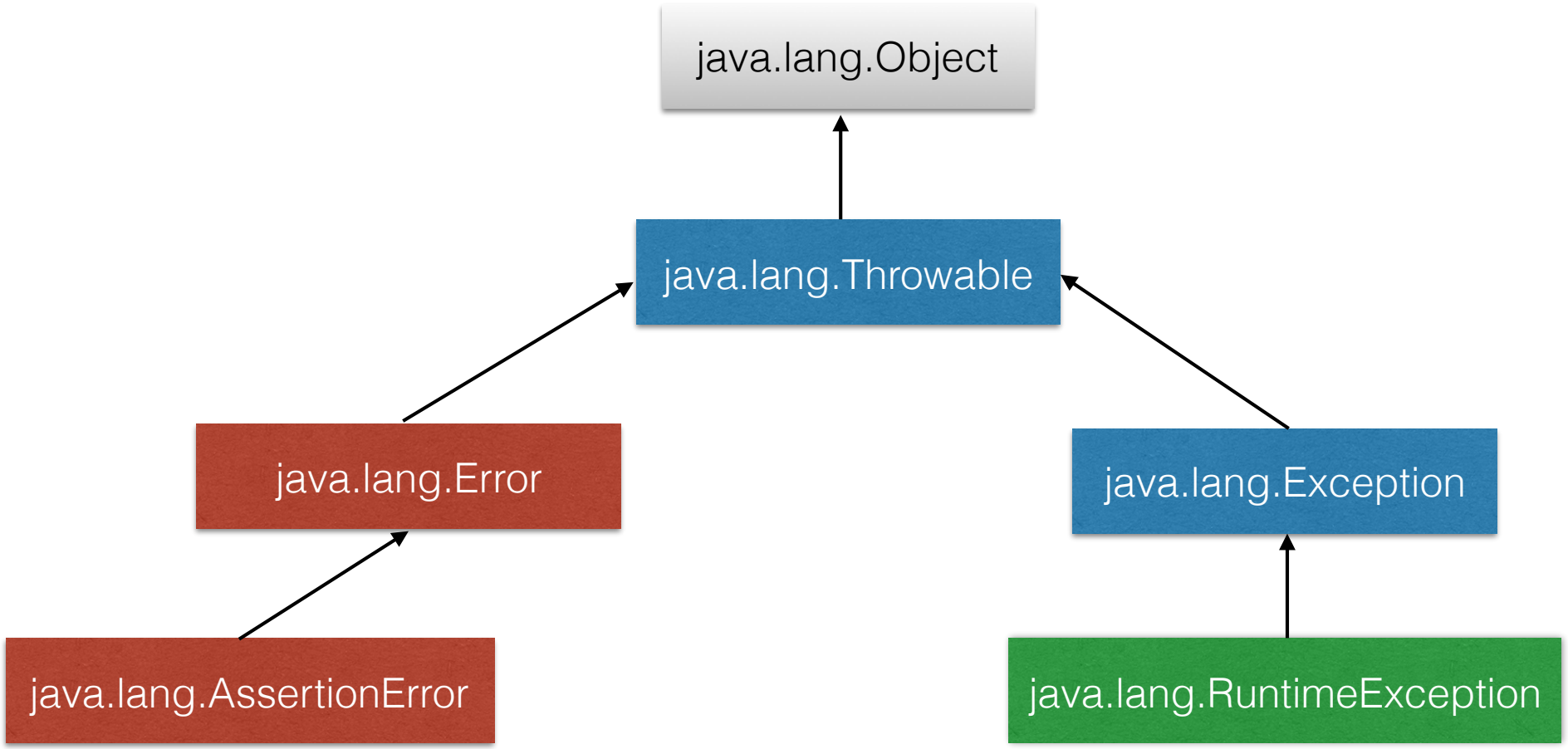
Keine Seiteneffekte!

```
assert(test());
//…

public boolean test() {
  intVar++;
  return true;
}
```

- Assert soll das Programm verlassen, wie es das vorgefunden hat

Exceptions


java.lang.Object
java.lang.Throwable
java.lang.Error
java.lang.Exception
java.lang.AssertionError
java.lang.RuntimeException

Exceptions

- Ausnahmen
- Informationen über einen Fehler
- Klassen, Basisklasse java.lang.Throwable
- 2 Arten:
  - checked (java.lang.Exceptions)
  - unchecked (java.lang.RuntimeException)

Multi-Catch

- Dürfen keine Subklassen voneinander sein

```
try {
  //…
}
catch(IOException | FileNotFoundException e) {
  //…
}
```

  - Compiler Fehler

- Typ von e ist nicht bekannt

- Reihenfolge innerhalb von catch ist unwichtig

Multi-Catch

```
try {
  //…
}
catch(IOException e) {
  e = new IOException();
}
```

- Bei Multi-Catch nicht möglich, da e final ist

Automatic Resource Management (try-with-resources)

- Auf jedes **try** muss **catch** oder **finally** folgen
- Bei **try-with-resources** nicht

```
try(FileInputStream x = new
FileInputStream("zeug.txt")) {
  //…
}
```

- Wird für einbinden von Streams und Ressourcen genutzt

AutoCloseable

- try-with-resources kann mit allen Klassen verwendet werden, die AutoCloseable implementieren
  `try(Integer i = 3) { } // Compiler Fehler`
- Closable implementiert AutoCloseable
- Ressourcen werden in umgekehrter Reihenfolge geschlossen
- close() sollte bei mehrfachen Aufruf (idempotent) keine Seiteneffekte verursachen.

Suppressed Exceptions

- Es ist möglich, dass während des try mehrere Exceptions geworfen wurden
- getSuppressed() der Exception liefert ein Array mit den unterdrückten Exceptions

Wichtige checked Exceptions

- java.text.ParseException
- java.io.IOException
- java.io.FileNotFoundException
- java.io.NotSerializableException
- java.sql.SQLException

Wichtige unchecked Exceptions

- ArithmeticException
- ArrayIndexOutOfBoundsException
- ClassCastException
- IllegalArgumentException
- NullPointerException
- NumberFormatException

Wichtige unchecked Exceptions

- java.lang.ArrayStoreException
- java.time.DateTimeException
- java.util.MissingResourceException
- java.lang.IllegalStateException
- UnsupportedOperationException

What will happen when you attempt to compile and run the following code?
(Assume that the code is compiled and run with assertions enabled.)

```
public class AssertTest
{
        private void methodA(int i)
        {
                assert i >= 0 : methodB();
                System.out.println(i);
        }

        private String methodB()
        {
                return "The value must not be negative";
        }

        public static void main(String [ ] args)
        {
                AssertTest test = new AssertTest();
                test.methodA(-10);
        }
}
```

Please select :

- A. It will print -10.
- B. It will result in AssertionError with the message -"The value must not be negative".
- C. The code will not compile.
- D. None of these

The AssertTest class defines a method, methodA(int i), as given below. Which of the following declarations of methodA(int i) would be valid in a subclass of AssertTest?
(Assume that the code is compiled successfully and runs with assertions enabled.)
Select three choices.

```
public class AssertTest
{
        public void methodA(int i) throws AssertionError
        {
                assert i < 1024 : "Invalid Value";
        }
}
```

Please select :

- [ ] A. public void methodA (int i)
- [ ] B. public void methodA (int i) throws Exception
- [ ] C. public void methodA (int i) throws Throwable
- [ ] D. public void methodA (int i) throws Error
- [ ] E. public void methodA (int i) throws RuntimeException
- [ ] F. Only option A is valid

What will happen when you attempt to compile and run the following code?

(Assume that the code is compiled and run with assertions enabled.)

```
1. public class AssertDemo
2. {
3.       private static boolean isValid(int i)
4.       {
5.             return 10%i > 0;
6.       }
7.
8.       public static void main(String [ ] args)
9.       {
10.            int i = 0;
11.            assert isValid(i) : "Invalid value";
12.            System.out.println(i);
13.      }
14. }
```

Please select :

- A. It will print-0
- B. Compilation error
- C. It will throw an AssertionError with message : Invalid value.
- D. It will throw an ArithmeticException.
- E. None of these.

Assume that assertions are enabled at both compiletime and runtime.

Given an assert statement -

assert Expression1 : Expression2;

Which of the following statements are true about Expression1?

Select two choices.

Please select :

- [ ] A. It always gets evaluated.
- [ ] B. It cannot be a method call.
- [ ] C. It can also be a method call to any method.
- [ ] D. It must be a boolean or Boolean expression.

Which of the following statements are true about the code snippet given below? (Assume that the code is compiled and run with assertions enabled.) Select three choices.

```
public class AssertDemo
{
        public static void main(String[ ] args)
        {
                int i = (int)(Math.random() * 100);
                assert i < 100 : "Out of range value";
                System.out.println(i);
        }
}
```

Please select :

- [ ] A. The assert condition i < 100 will never be evaluated to false.
- [ ] B. The code will run and print a value between 1 and 100, both inclusive.
- [ ] C. The code will run and print a value between 0 and 100, both inclusive.
- [ ] D. The code will run and print a value between 0 and 99, both inclusive.
- [ ] E. The assert condition i < 100 may be evaluated to false.
- [ ] F. The assert statement effectively tests the output range of Math.random().

Which of the following is the most appropriate code snippet that can be inserted at line 3?

```
1. public void setFrequency(int frequency)
2. {
3.       // What should be inserted here?
4.       this.frequency = frequency;
5. }
```

Please select :

- A. assert ((0 < frequency) && (frequency <=60));
- B. assert ((0 < frequency) && (frequency <=60)) : "Frequency out of range";
- C. if(frequency <= 0 || frequency > 60)
  throw new IllegalArgumentException("Frequency out of range");
- D. if(frequency <= 0 || frequency > 60)
  throw new Exception("Frequency out of range");
- E. None of these

What will happen when you attempt to compile and run the following code? (Assume that the code is compiled and run with assertions enabled.)

```
1. public class AssertTest
2. {
3.       public static void main(String [ ] args)
4.       {
5.             for ( int i = 0; i < 10; i++ )
6.             {
7.                   try
8.                   {
9.                         assert i%2==0 : i--;
10.                        System.out.println("Even number : " + i);
11.                  }
12.                  catch(AssertionError ae)
13.                  {
14.                        System.out.println("Odd number : " + ++i);
15.                  }
16.            }
17.      }
18. }
```

Please select :

- A. It will print odd and even numbers from 0 to 9 correctly (0 even and 1 odd).
- B. It will print odd and even numbers from 0 to 9 incorrectly (0 odd and 1 even).
- C. Compilation error at line 9.
- D. Compilation error at line 10.
- E. It will result in an infinite loop.

In the following code, which lines do not use assertions appropriately?

Select two choices.

```
1. public class AssertDemo
2. {
3.       public static Boolean message()
4.       {
5.             System.out.println("Error!!!");
6.             return new Boolean("true");
7.       }
8.       public static void main(String [ ] args)
9.       {
10.            boolean b = false;
11.            assert b : message();
12.            int x = -5;
13.            assert (x+1) > 0;
14.            assert x > 0 : (x = x * -1);
15.            assert !b;
16.            while (true)
17.            {
18.                  System.out.println(x);
19.                  assert x-- > 0;
20.            }
21.      }
22. }
```

Please select :

- [ ] A. Line 11
- [ ] B. Line 13
- [ ] C. Line 14
- [ ] D. Line 15

Given Code:

```
1.  import java.io.*;
2.  class Writer2 {
3.          public static void main(String args [ ] ) {
4.                  char[ ] in = new char[50];
5.                  int size = 0;
6.                  File file= new File("fileWriter2.txt");
7.                  try(FileWriter fw = new FileWriter(file), FileReader fr = new FileReader(file)) {
8.                          fw.write("howdy\nfolks\n");
9.                          size=fr.read(in);
10.                         System.out.print(size + " ");
11.                         for(char c: in)
12.                         System.out.print(c);
13.                   } catch(IOException e) { }
14.         }
15.  }
```

What will be the result?

Please select :

- A. 12 howdy
  folks
- B. Compilation fails
- C. Compiles suucessfuly with no output.
- D. Exception is thrown at runtime

Given Code:

```
1.  import java.io.BufferedReader;
2.  import java.io.FileReader;
3.  public class Reader {
4.          public static void main(String [ ] args) {
5.                  try (String r = new String("temp.txt"))
6.                  {
7.                          System.out.println("Hello World");
8.                  } catch (Exception e) {
9.                          System.out.println("Good Bye");
10.                 }
11.         }
12.  }
```

What will be the output?

Please select :

- A. Hello World
- B. Good Bye
- C. Compile time error
- D. Code will run successfully without any output

Given Code:

```
1.  package com.test;
2.  class ResourceException extends Exception { }
3.  class CloseException extends Exception { }
4.  public class ExceptionThrower implements AutoCloseable {
5.          public void getresource() throws Exception {
6.                  throw new ResourceException(); }
7.          public void close() throws Exception {
8.                  throw new CloseException(); }
9.          public static void main(String [ ] args) throws Exception {
10.                 try (ExceptionThrower exceptionClass = new ExceptionThrower()) {
11.                         exceptionClass.getresource();
12.                 } catch (Exception e) {
13.                         System.out.println(e.toString());
14.                         System.out.println(e.getSuppressed()[0].toString());
15.                 }
16.         }
17. }
```

What will be the output?

Please select :

- A. Code will run without any output
- B. com.test.ResourceException
- C. com.test.CloseException
- D. . com.test.ResourceException
  com.test.CloseException
- E. com.test.CloseException
  com.test.ResourceException

Given :

```
1.  import java.io.*;
2.  public class Reader {
3.          public static void main(String [ ] args) {
4.                  FileIReader fr;
5.                  try (fr = new FileReader("java.txt"))
6.                 {
7.                          System.out.println("Java Programming");
8.                  } catch (Exception e) {
9.                          System.out.println("File do not exist");}
10.         }
11.  }
```

What will be the output of the above code fragment?

Please select :

- A. Java Programming
- B. File do not exist
- C. Java Programming
  File do not exist
- D. Compilation Fails

```
1. public class A {
2.       public void methodA() {
3.             B bObj = new B();
4.             bObj.methodB();
5.             //more code here
6.       }
7. }
```

```
1. public class B {
2.       public void methodB() {
3.            C cObj = new C();
4.            cObj.methodC();
5.            //more code here
6.       }
7. }
```

```
1. public class C {
2.       public void methodC() {
3.             //more code here
4.       }
5. }
```

```
25. try {
26.       A aObj = new A();
27.       aObj.methodA();
28. } catch (Exception e) {
29.       System.out.print("You got an exception!");
30. }
```

Which two statements are true if a NullPointerException is thrown on line 3 of class C?
(Select two options.)

Please select :

- [ ] A. The application will crash.
- [ ] B. The code on line 29 will be executed.
- [ ] C. The code on line 5 of class A will execute.
- [ ] D. The code on line 5 of class B will execute.
- [ ] E. The exception will be propagated back to line 27.

Given the following code:

```
11. static void testStrings() {
12.       try {
13.             String nullString = null;
14.             System.out.print(nullString.toString() + " ");
15.       }
16.       finally { System.out.print("finally "); }
17. }
18. public static void main(String [ ] args) {
19.        try { testStrings(); }
20.        catch (Exception ex) { System.out.print("exception"); }
21. }
```

What is the result?

Please select :

- A. null
- B. finally
- C. null finally
- D. Compilation fails.
- E. finally exception

Given the following code:

```
11. static void testMethod() throws RuntimeException {
12.         try {
13.                 System.out.print("test method. ");
14.                 throw new RuntimeException();
15.         }
16.         catch (Exception ex) { System.out.print("exception. "); }
17.  }
18. public static void main(String [ ] args) {
19.         try { testMethod(); }
20.         catch (RuntimeException rte) { System.out.print("run time exception. "); }
21.         System.out.print("program ends. ");
22. }
```

What is the result?

Please select :

- A. test method. program ends.
- B. Compilation fails.
- C. test method. run time exception. program ends.
- D. test method. exception. program ends.
- E. A Throwable is thrown by main at runtime.

Suppose MyException (which is a subclass of Exception) should be thrown if condition (which is a boolean variable) is true, which statements would you insert? Select two choices.

```
1. public aMethod
2. {
3.         if (condition) {
4.
5.         }
6. }
```

Please select :

- [ ] A. throw new Exception() at line 1
- [ ] B. throw new MyException() at line 4
- [ ] C. throw new MyException() at line 6
- [ ] D. throws new Exception() at line 3
- [ ] E. throws MyException at line 1

Given the following code:

```
1. class UserDefinedException extends Exception { }
2. class Hello {
3.         public String sayHi(String firstName) throws UserDefinedException {
4.                 if (firstName == null) throw new UserDefinedException();
5.                     return "Hi " + firstName;
6.         }
7. }
8. public class TestHello {
9.          public static void main(String[ ] args) {
10.                 new Hello().sayHi("Rambo");
11.         }
12. }
```

Which statement is true?

Please select :

- A. Compilation succeeds.
- B. Class Hello does not compile.
- C. The method declared on line 9 cannot be modified to throw UserDefinedException.
- D. TestHello compiles if line 10 is enclosed in a try/catch block that catches UserDefinedException.

Given the following code:

```
1. public class TestAssertions {
2.       public static void main(String [ ] args) {
3.             boolean assertsEnabled = false;
4.             assert (assertsEnabled) : assertsEnabled = true;
5.             if (assertsEnabled) {
6.                   System.out.println("asserting");
7.             }
8.       }
9. }
```

If class TestAssertions is invoked twice, the first time without assertions enabled, and the second time with assertions enabled, what is the result?

Please select :

- A. no output
- B. no output
  asserting
- C. asserting
- D. no output
  An AssertionError is thrown.
- E. asserting
  An AssertionError is thrown.

Given the following code:

```
1. public class Horse {
2.       public static void main(String [ ] args) {
3.             boolean assertionsOn = false;
4.             assert (assertionsOn) : assertionsOn = true;
5.             if (assertionsOn) {
6.                   System.out.println("assertions are enabled");
7.             }
8.       }
9. }
```

If class Horse is invoked twice, the first time without assertions enabled, and the second time with assertions enabled, what is the result?

Please select :

- A. no output
- B. no output
  assertions are enabled
- C. assertions are enabled
- D. no output
  An AssertionError is thrown.
- E. assertions are enabled
  An AssertionError is thrown.

What will be the result of an attempt to compile and run the following program?

```
class Test
{
	void f() throws NullPointerException
	{
		throw new RuntimeException();
	}
	public static void main (String[ ] args) throws Exception
	{
		Test t=new Test();
		try
		{
			t.f();
		}
		catch (Exception e)
		{
			System.out.println("catch");
		}
		finally
		{
			System.out.println("finally");
			throw e;
		}
	}
}
```

Please select :

- A. Code does not compile because f() does not declare that it throws RuntimeException.
- B. Compiles and prints "catch finally".
- C. Compiles and prints "finally".
- D. Compiles and prints "catch".
- E. Compiles and runs without output.
- F. None of the above.

Gien the following code :

```
1. class Ex6 {
2.       public static void main(String args [ ]) {
3.             try {
4.                   new Ex6().meth();
5.             } catch(ArithmeticException e) {
6.                   System.out.print("Arithmetic ");
7.             } catch(Exception e) {
8.                   System.out.print("Exception ");
9.             }
10.            System.out.print("Now ");
11.            finally {
12.                  System.out.print("final");
13.            }
14.      }
15.
16.      public void meth()throws ArithmeticException {
17.            for ( int x = 0; x<5; x++ ) {
18.                  int y = (int)5/x;
19.                  System.out.print(x);
20.            }
21.      }
22. }
```

What is the output?

Please select :

- A. Arithmetic final
- B. Exception final
- C. Arithmetic Now final
- D. Exception Now final
- E. Compilation fails.
- F. Arithmetic

Gien the following code :

```
1.  class Ex6 {
2.      public static void main(String args [ ] ) {
3.          try {
4.              throw new ArithmeticException();
5.          } catch(Exception e) {
6.              System.out.print("Exception 1 ");
7.          } finally {
8.              try {
9.                  new Ex6().meth();
10.             } catch(Exception e) {
11.                 System.out.print("Exception 2 ");
12.             }
13.         }
14.     }
15.
16.     public void meth() {
17.         for ( int x = 0; x<5; x++ ) {
18.             int y = (int)5/x;
19.             System.out.print(x);
20.         }
21.     }
22. }
```

What is the output?

Please select :

- A. Exception 1
- B. Exception 1 Exception 2
- C. Exception 2
- D. Compilation fails.
- E. Exception 1 1234

Given the following code :

```
1.  class Ex6 {
2.          Integer I;
3.          public static void main(String args [ ] ) {
4.                  String s;
5.
6.                  try {
7.                          s = new Ex6().I.toString();
8.                  } catch(NumberFormatException e) {
9.                          System.out.print("NumberFormat ");
10.                 } catch(ClassCastException e) {
11.                         System.out.print("ClassCast ");
12.                 } catch(NullPointerException e) {
13.                         System.out.print("Nullpointer ");
14.                 } catch(Exception e) {
15.                         System.out.print("Exception ");
16.                 } finally {
17.                         System.out.print("Finally");
18.                 }
19.         }
20. }
```

What is the output?

Please select :

- A. NumberFormat ClassCast Nullpointer Exception Finally
- B. Nullpointer Exception Finally
- C. Nullpointer Finally
- D. NumberFormat Finally
- E. ClassCast Finally
- F. Exception Finally

Given the following code :

```
1.  class ExTest {
2.        Integer I;
3.        public static void main(String args [ ] ) {
4.              String s;
5.              try {
6.                    s = new Ex6().I.toString();
7.              } finally {
8.
9.              try {
10.                   int i = Integer.parseInt(args[0]);
11.             } catch(NumberFormatException E) {
12.                   System.out.print("NumberFormat ");
13.             } finally {
14.                   System.out.print("Fin2 ");
15.             }
16.                   System.out.print("Fin1 ");
17.             }
18.       }
19. }
```

And the given command line invocation is java ExTest 10
What is the output?

Please select :

- A. NumberFormat Fin1 Fin2
- B. NumberFormat Fin1
- C. NumberFormat Fin2
- D. Fin1 Fin2
- E. Fin1
- F. Fin2 Fin1 followed by uncaught exception

Given the following code :

```
1.  class ExTest {
2.        Integer I;
3.        public static void main(String args [ ] ) {
4.              String s;
5.              try {
6.                    s = new Ex6().I.toString();
7.              } finally {
8.
9.                    try {
10.                         int i = Integer.parseInt(args[0]);
11.                   } catch(NumberFormatException E) {
12.                         System.out.print("NumberFormat ");
13.                   } finally {
14.                         System.out.print("Fin2 ");
15.                   }
16.             System.out.print("Fin1 ");
17.             }
18.       }
19. }
```

And the given command line invocation is java ExTest A
What is the output?

Please select :

- A. NumberFormat Fin2 Fin1
- B. NumberFormat Fin2
- C. NumberFormat Fin2 followed by uncaught exception
- D. Fin2 Fin1 followed by uncaught exception
- E. NumberFormat Fin2 Fin1 followed by uncaught exception
- F. Fin2 Fin1 followed by uncaught exception

Given the following code :

```
1.  class TooSmallException extends Exception {
2.  }
3.
4.  class Divider {
5.          public double divide(double d1, double d2) {
6.                  if ( d1<0.01 || d2<0.01 ) {
7.                          throw new TooSmallException();
8.                  } else {
9.                          return (d1/d2);
10.                 }
11.         }
12. }
13.
14. public class ETest {
15.          public static void main(String [ ] args) {
16.                  Divider d = new Divider();
17.
18.                  double i = Double.parseDouble(args[0]);
19.                  double j = Double.parseDouble(args[1]);
20.
21.                  double ans = d.divide(i,j);
22.                  System.out.print(ans);
23.          }
24. }
```

Which of the statement is true?

Please select :

- A. Compilation succeeds and if we pass 1 and 2 as “args[0]” and “args[1]” the output will be an uncaught TooSmallException.
- B. Compilation succeeds and if we pass 0.001 and 0.2 as “args[0]” and “args[1]” the output will be an uncaught TooSmallException.
- C. Compilation fails due to error on line 7 as there is no Exception class called as “TooSmallException” in java.
- D. Compilation fails as “TooSmallException” is checked Exception and we didn’t declare it or handle it.
- E. Compilation succeeds and if we pass “A” and “B” as “args[0]” and “args[1]” 0.9848484848484849 will be printed as the output.(Note ASII value of ‘A’ is 65 and ‘B’ is 66 in decimals.)

```
1.  import java.io.*;
2.  class NotInRangeException extends IOException {
3.  }
4.
5.  class Employee {
6.          int age;
7.          public boolean setAge(int ae)throws NotInRangeException {
8.                  if ( ae < 18 || ae > 60 ) {
9.                          throw new NotInRangeException();
10.                 } else {
11.                         age = ae;
12.                         return true;
13.                 }
14.         }
15. }
16. public class ETest4 {
17.          public static void main(String [ ] args) {
18.                  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
19.                  Employee d = new Employee();
20.                  System.out.print("Enter the age : ");
21.                  try {
22.                          br.readLine();
23.                          d.setAge(5);
24.                  } catch(IOException E) {
25.                          System.out.print(E);
26.                  } catch(NotInRangeException E) {
27.                          System.out.print(E);
28.                  }
29.          }
30.  }
```

Which option is true?

Please select :

- A. There is no constructor in java.lang.IOException class which takes a noting so code fails to compile.
- B. We can't create custom exception by extending java.io.IOException class so code fails to compile.
- C. Compilation fails due to error on line 26.
- D. Compilation succeeds and if we enter a number less than 18, a "NotInRangeException" will be thrown.
- E. The "NotInRangeException" is unchecked exception.

```
1.  import java.io.*;
2.  class NegativeException extends RuntimeException {
3.  }
4.
5.  class NotEnoughException extends NegativeException {
6.  }
7.
8.  class Lone {
9.          int amount;
10.         public boolean setAmount(int am)throws NotEnoughException {
11.
12.                 if( am < 0 ) {
13.                         throw new NegativeException();
14.                 } else if ( am < 500 ) {
15.                         throw new NotEnoughException();
16.                 } else {
17.                         amount = am;
18.                         return true;
19.                 }
20.         }
21. }
22. public class ETest5 {
23.
24.          public static void main(String [ ] args) {
25.                  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
26.                   Lone d = new Lone();
27.                   System.out.print("Enter the amount : ");
28.                   try {
29.                           int i = Integer.parseInt(br.readLine());
30.                           d.setAmount(i);
31.                   } catch(NegativeException E) {
32.                           System.out.print(E);
33.                   } catch(NotEnoughException E) {
34.                           System.out.print(E);
35.                   } catch(Exception E) {
36.                           System.out.print(E);
37.                   }
38.           System.out.print(d.amount);
39.       }
40. }
```

Which option is true?

Please select :

- A. Both custom exceptions are checked exceptions
- B. We can't create custom exception by extending java.lang.RunimeException class so code fails to compile.
- C. Compilation fails due to an error on line 33.
- D. Compilation succeeds.
- E. Only the "NegativeException" is unchecked exceptions

Which option is true?

Please select :

- A. We can't create custom exception using subclass of java.lang.RuntimeException class.
- B. We can create custom unchecked exception by extending java.lang.Exception class.
- C. We can create custom checked exception by extending java.lang.RuntimeException class.
- D. We can create custom checked exception using subclass of java.lang.Exception class.
- E. None ofAbove.

What are the times we may have to choose create custom exception?
(Choose the most suitable answers)

I. You need an exception type which is already in java.
II. You need an exception type which is not in java.
III. You need an exception type which is not in java but you can manage it by using other one.

Please select :

- A. Only I
- B. Only I and III
- C. Only III
- D. Only II
- E. All

Which options are incorrect?

I. Java.lang.Exception class has only 4 constructors.
II. One constructor of Java.lang.Exception takes a String as the parameter.
III. If we invoke the “Exception()” constructor of the Java.lang.Exception class,We can Construct a new exception with the specified detail message..
IV. If we invoke the “Exception(String message)” constructor of the Java.lang.Exception class,We can Construct a new exception with the specified detail message.

Please select :

- A. Only I & II
- B. Only II and IV
- C. Only II and III
- D. Only III
- E. Only I and III.
- F. All

Given :

```
1. public class Whizlab {
2.         public static void main(String[] args) {
3.                 assert args == null : "Null ";
4.                 System.out.print("OK");
5.         }
6. }
```

What would be the output if the assertion is enabled?

Please select :

- A. Good
- B. Null
- C. Null OK
- D. An AssertionError with massage Null
- E. Compilation fails.

Given :

```
1. import java.io.*;
2. class NegativeException extends ClassCastException {
3.         NegativeException(){   super("Negative Value entered, Positive value expected"); }
4.         NegativeException(String s){  super(s);  }
5. }
6. class NotEnoughException extends NegativeException{
7.         NotEnoughException(){  super("Less than minium");  }
8. }
9.
10. class Loan{
11.         int amount;
12.         public boolean setAmount(int am){
13.                 if( am < 0){
14.                         throw new NegativeException();
15.                 }else if( am < 500){
16.                         throw new NotEnoughException();
17.                 }else{
18.                         amount = am;
19.                         return true;
20.                 }
21.         }
22. }
23. public class Whizlab{
24.         public static void main(String [] args){
25.                 BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
26.                 Loan l = new Loan();
27.                 System.out.print("Enter the amount : ");
28.                 try{
29.                         int i = Integer.parseInt(br.readLine());
30.                         l.setAmount(i);
31.                 }catch(IOException E){
32.                         System.out.print(E);
33.                 }
34.                 System.out.print(l.amount);
35.         }
36. }
```

Which is true?

Please select :

- A. Compilation fails as we can't create a custom exception by extending another custom exception.
- B. Compilation fails as we can't create custom exception by extending a subclass of java.lang.RuntimeException class.
- C. Compilation fails as we didn't declare or handle "NegativeException" and "NotEnoughException".
- D. Compilation succeeds and if we enter a number less than 0, a "NegativeException" will be thrown and it'll look like "Exception in thread "main" NegativeException:"
- E. Compilation succeeds and if we enter a number less than 500, a "NotEnoughException" will be thrown and it'll look like "Exception in thread "main" NotEnoughException: Less than minium".

Consider following code fragment :

```
catch (NullPointerException e){
        //codes
 }catch(ArithmeticException ex){
        //codes
 }
```

Which of the following can be used to replace above code?

Please select :

- A. catch (NullPointerException e1 & ArithmeticException e2){ /*codes */}
- B. catch (NullPointerException | ArithmeticException e){ /*codes */}
- C. catch (NullPointerException ex | ArithmeticException ex2){ /*codes */}
- D. catch (NullPointerException ex | ArithmeticException ex){ /*codes */}
- E. catch (NullPointerException || ArithmeticException e){ /*codes */}

Given :

```
1.      class Whizlab{
2.
3.              public static void main(String args[]){
4.                      try{
5.                              System.out.println(args[0]);
6.                      }catch (ArrayIndexOutOfBoundsException | ArithmeticException | NullPointerException e){
7.                              if(e instanceof ArrayIndexOutOfBoundsException){
8.                                      e = new ArrayIndexOutOfBoundsException("Out of bounds");
9.                              }else if(e instanceof NullPointerException){
10.                                     e = new NullPointerException("Null Value");
11.                             }else{
12.                                     e = new ArithmeticException("Arithmetic");
13.                             }
14.                             System.out.println(e.getMessage());
15.                     }
16.             }
17.  }
```

What is the output?

Please select :

- A. Null
- B. Null Value
- C. Arithmetic
- D. Out of bounds
- E. Compilation fails.

Given :

```
1. public class Whizlab{
2.     public static void main(String[] args) {
3.         try(Resource r = new Resource()){
4.                 System.out.print("1");
5.                 throw new RuntimeException();
6.         }
7.     }
8. }
9. class Resource implements AutoCloseable{
10.                @Override
11.                public void close() {
12.                        System.out.println("closed");
13.                }
14. }
```

What is the output?

Please select :

- A. 1
- B. closed
- C. 1closed
- D. 1closed followed by an exception
- E. None.

Given :

```
1.  public class Whizlab{
2.      public static void main(String[] args) {
3.          int j = 9;
4.          assert(++j > 7): "Error";
5.          assert(j==12): j;
6.          assert(++j > 8): System.out.println(j);
7.          assert(j==12): new Whizlab();
8.      }
9.  }
```

What is the output?

Please select :

- A. 8
- B. 9
- C. Compilation fails due to line 5.
- D. Compilation fails due to line 6.
- E. Compilation fails due to multiple errors.

Given :

```
1.      class Whizlab{
2.          public static void main(String args[]){
3.                  new Whizlab().meth();
4.                  }
5.
6.          public void meth()throws Exception{
7.                  for(int x=0;x>5;x++)
8.                          System.out.print(x);
9.                  }
10.   }
```

What is the output?

Please select :

- A. 01234
- B. An exception is thrown at runtime.
- C. Code will cause a never ending loop.
- D. Compilation fails.
- E. No output.