Emre KARAGÖZ

Assignment-8
Handling Java Exceptions

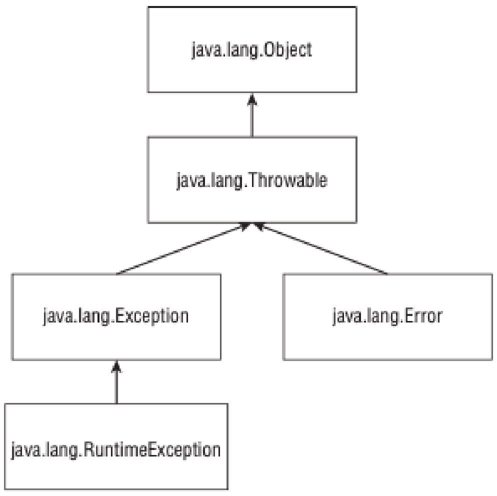
Q1) Option D

The code does not compile because a try statement needs a catch block or a finally block. **Option D** is the answer.

Q2) Option B

When they were used together ***"try", "catch", "finally"*** is the correct order.

Q3) Option D



The figure above shows ***"java.lang"*** classes inheritance model properly. **Option D** is the answer.

Q4) Option A

"`Exception`"s including "`RuntimeException`"s can be caught in Java. "Error" means something went so horribly wrong that your program should not attempt to recover from it. Because of this an "Error" is recommended not to caught. **Option A** is the answer. Notice that there is not a class called "`CheckedExcetion`" (Option B) in Java.

Q5) Option D

"`score`" variable declared in try block and it is only accessible in that scope. Since compiler cannot access it catch and finally blocks, the code does not compile. Option D is the answer.

Q6) Option B

Option A: `ClassCastException`

"`RuntimeException`" exception that is thrown by the JVM when an attempt is made to cast an exception to a subclass of which it is not an instance.

Option B: `IOException`

This class is the general class of exceptions produced by failed or interrupted I/O operations. These exceptions are ***checked exceptions*** and they must be declared or handle.

Option C: `ArrayIndexOutOfBoundsException`

A "`RuntimeException`" exception that is thrown by the JVM when code uses an illegal index to access an array

Option D: `IllegalArgumentException`

A "`RuntimeException`"exception that is thrown by the programmer to indicate that a method has been passed an illegal or inappropriate argument

Q07) Option A

The ***"throws"*** keyword is used in method declarations, while the ***"throw"*** keyword is used to throw an exception to surrounding process. **Option A** is the answer.

Q08-) Option B

Since "`IOException`" is a subclass of "Exception", the catch block for "`IOExceptions`" must appear before the catch block for "Exception" in order to prevent the code from a compiler error about unreachable code.

Q09-) Option C

The code does not compile because there is not a defined exception called "t".

Q10-) Option C

The code does not compile because of line p3. Method "`DrawBridge`" throws an "`Exception`" but main method does not declare or handle the exception, but it must declare or handle. **Option C** is the answer.

Q11-) Option B

"NullPointerException" and "ArithmeticException" are subclasses of "RuntimeException", which are unchecked exceptions and unchecked exceptions are not required to be handled or declared in the method in which they are thrown such as their superclass "RuntimeException". Exception is a checked exception that can be thrown by the programmer or by the JVM and it must be handled or declared. **Option B** is the answer.

Q12-) Option A

The code compiles without an error and prints "1345". First, try block prints "1" and throws "ClassCastException". The first catch block is skipped because "ClassCastException" is not a subclass of "ArrayOutOfBoundsException".
Since "ClassCastException" is a subclass of "Throwable", the second catch block is executed and prints "3". Then, "finally" block is executed and prints 4. Lastly, "5" is printed.

Q13-) Option C

Every line of the "finally" block is not guaranteed to be executed. "finally" block can throw an "Exception" and if it throws, remaining lines of finally block are not executed. So, **Options A and D** are false. **Option B** is also false because the finally block always executes regardless of the catch block. **Option C** is the answer. The "finally" statement requires brackets.

Q14-) Option C

The code does not compile because catch blocks are in wrong order. Since "FileNotFoundException" is subclass of "IOException" If the more specific "FileNotFoundException" is thrown, the catch block for "IOException" runs. This means there is no way for the second catch block to ever run. Compiler tells us there is an unreachable catch block. **Option C** is the answer.

Q15-) Option C

"catch" and ***"finally"*** keywords are required with a try statement. They can be used both but, one of them is sufficent to compile the code. ***"finalize"*** is not a keyword. finalize() is the method of Object class. This method is called just before an object is garbage collected. finalize() method overrides to dispose system resources, perform clean-up activities and minimize memory leaks. **Option C** is the answer.

Q16-) Option B

An exception is Java's way of saying, "I give up. I don't know what to do right now. You deal with it." Exceptions are often used when things "go wrong" or deviate from the expected path. So, **Option A** is true. **Option C** is also true because before compile the code, some exceptions can be avoid programmatically. Exception handling may recover from unexpected problems and prevents application from the terminating. So, **Option D** is true and **Option B** is false and it is the answer.

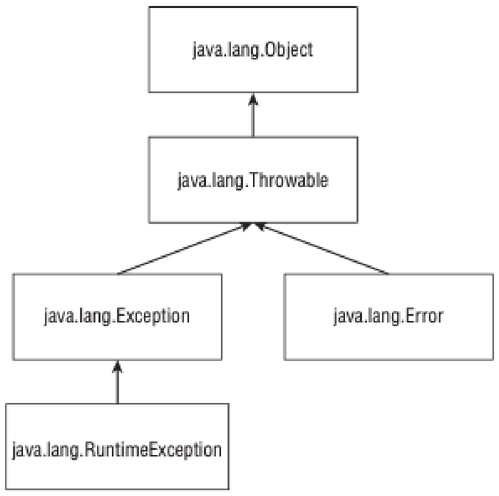
Q17-) Option D

The code does not compile because Java does not allow to use catch block with parenthesis. It must be used with brackets.

Q18-) Option B

An overridden method can throw the same exception or a narrower one than the exception thrown by the method that it inherits. But it does not have to throw an exception. Since `"Exception"` is broader than `"PrintException",` the method signature in Option B is not allowed in a class which implements interface `"Printer".` **Option B** is the answer.

Q19-) Option D



As shown above; `"Throwable", "Exception", "RuntimeException"` classes are belongs to "java.lang" package which is a special package in Java. This package is special in that it is automatically imported. So, no import statement is needed. **Option D** is the answer.

Q20-) Option C

The code does not compile due to line “g3”. An exception is required to catch by a "`catch`" statement, but it is missing. **Option C** is the answer.

Q21-) Option B

A program must handle or declare "checked exceptions", but should never handle "java.lang.Error". Checked exceptions cause compiler error when they are not declared handled. Option B is the answer.

Q22-) Option B

While checked exceptions must be declared or handle, unchecked exceptions are not required to be declared or handle. Since "`ClassCastException`" is a checked exception, it must be declared in line q1 in order to compile line q2 or must be handled in finally block. Because of these reasons the code does not compile. **Option B** is the answer.

Q23-) Option A

If an exception matches two or more catch blocks, the first one that matches is executed. Moreover, a rule exists for the order of the catch blocks. Java looks at them in the order they appear. If it is impossible for one of the catch blocks to be executed, a compiler error about ***unreachable code*** occurs. This happens when a superclass is caught before a subclass. **Option A** is the answer.

Q24-) Option C

The method "`compute()`" is called by main method and this call prevents the code from compiling. Even though the method "`compute()`" does not throw a checked exception, declaration of it tells it may throw. In order to compile this code, we must declare or handle a checked exception in main method. Because checked exceptions must be declared or handle.

Q25-) Option D

```
private Boolean[] list = {true, false}; // ArrayIndexOutOfBoundsException is thrown
private Boolean[] list = (Boolean[]) new Object(); // ClassCastException is thrown
private Boolean[] list = null; // NullPointerException
```

All these exceptions may be thrown depending on the value that omitted. **Option D** is the answer.

Q26-) Option B

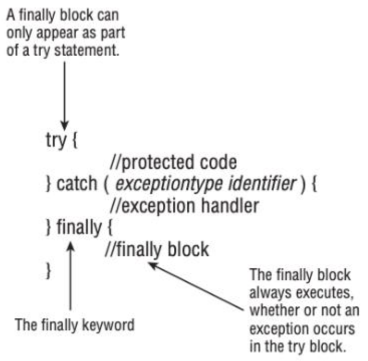
`StackOverflowError:` Thrown by the JVM when a method calls itself too many times (this is called infinite recursion because the method typically calls itself without end).

`NullPointerException:` Thrown by the JVM when there is a null reference where an object is required.

A "**`StackOverflowError`**" occurs when a program recurses too deeply into an infinite loop, while a "**`NullPointerException`**" occurs when a reference to a nonexistent object is acted upon. Option D is the answer.

Q27-) -

Q28-) Option D



```
try {
        //protected code
} catch ( exceptiontype identifier ) {
        //exception handler
} finally {
        //finally block
}
```

A finally block can only appear as part of a try statement.
The finally keyword
The finally block always executes, whether or not an exception occurs in the try block.

This is the syntax of a try statement with finally. The code does not compile because the catch and finally blocks are in the wrong order. Finally block must be used after the catch block. **Option D** is the answer. If the order corrected, the code outputs "`Finished! Joyce Hopper`".

Q29-) Option A

A try statement has ***"zero or one" finally*** block(s) and ***"zero or more"*** catch blocks. However, if there is no catch and finally statement, the code does not compile. **Option A** is the answer.

Q30-) Option D

The code compiles but throws an "`ArithmeticException`" at runtime because default value of variable count is "0". "`ArithmeticException`" is thrown by the JVM when code attempts to divide by zero. **Option D** is the answer.

Q31-) Option B

When an new exception is thrown in a catch block or finally block that will propagate out of that block, then the current exception will be aborted (and forgotten) as the new exception is propagated outward. If both the catch and finally blocks throw an exception, the one from the finally block is propagated to the caller. Caller sees the exception from the finally block. **Option B** is the answer.

Q32-) Option A

The code does not compile because of line m1. **Option A** is the answer. Exceptions must be declared by `"throws"` keyword, not by `"throw"` keyword. On the other hand, `"RuntimeException"` is an unchecked exception so it does not have to declare or handle.

Q33-) Option A

`"ClassCastException"` is thrown because `"Exception"` cannot be cast to `"RuntimeException"`.

Q34-) Option C

Since all exceptions inherit from class `"Throwable"` in Java, **Option C** is the answer.

Q35-) Option B

If both omitted values are valid and not null the code compiles without an issue. The "finally" block always executes, whether or not an exception occurs in the try block. So, all possible outputs must start with "Posted". Second statement can be possible output, but first statement cannot. **Option B** is the answer.

Q36-) Option A

Since `"ClassCastException"` is a subclass of `"RuntimeException"`, the catch block for `"ClassCastException"` must appear before the catch block for `"RuntimeException"`.

Q37-) Option C

Passing invalid data to a method is a good case to use an exception. There is an exception that called `"IllegalArgumentException"` which is thrown by the programmer to indicate that a method has been passed an illegal or inappropriate argument. If the code does not compile, it does not throw exceptions. In order to throw an exception, it must be compile firstly. I think completing a method sooner than expected is not a good choice. **Option C** is the answer.

Q38-) Option C

As explained in Q18 an overridden method can throw the same exception or a narrower one than the exception thrown by the method that it inherits. Since `"Exception"` is superclass of `"RuntimeException"`, the code does not compile. In order to compile this code `"Exception″` and "RuntimeException" must be switched. **Option C** is the answer.

Q39-) Option D

Since a `〝NullPointerException″` is an unchecked exception, it can be handled with a try-catch block or ignored altogether by the surrounding method. **Option D** is the answer.

Q40-) Option D

The code does not compile because an object is needed to throw. In order to create a `〝RuntimeException″` object, “new” keyword is required. **Option D** is the answer.

Q41-) Option C

The finally block always executes whether or not an exception occurs in the try and catch blockes and it throws a `〝RuntimeException″` which is printed in the stack trace at runtime. **Option C** is the answer.

Q42-) Option D

```
public void catchBall() throws OutOfBoundsException
```

In order to override a method correctly, return type must be covariant with void and the method only can throws `〝OutOfBoundsException″` or a narrower one. Since all these methods return “int” which is not a covariant with void, **Option D** is the answer.

Q43-) Option D

The code does not compile because catch block is missing a variable name. **Option D** is the answer. In order to compile this code, we must add a variable name. If we do this, the code compiles and throws `〝IllegalArgumentException″.`

Q44-) Option D

The code compiles, but a stack trace is printed at runtime. Eventually “finally” block is executed and throws a `〝RuntimeException″.` **Option D** is the answer.

Q45-) Option C

"`IllegalArgumentException`" and "`ClassCastException`" are subclasses of "`RuntimeException`" but none of them is inherited from the other one. So, the catch blocks for these two exception types can be declared in any order. **Option C** is the answer.

Q46-) Option D

Since "`RuntimeException`" is not an interface, it cannot be implemented, and code does not compile. **Option D** is the answer. "`RuntimeException`" is a class and subclasses such as "Problem" can be created from it by using ***"extends"*** keyword. If we do this the code compiles and prints "Problem?Fixed!"

Q47-) Option D

The code does not compile because an exception cannot be thrown by using "throws" keyword. "throws" keyword is used in order to declare an exception, "throw" keyword must be used for throwing. **Option D** is the answer.

Q48-) –

Q49-) Option C

The code does not compile due to line "z1". Since variable "e" is already defined in the catch block, we cannot define another variable with same name. **Option C** is the answer.

Q50-) Option D
The code does not compile due to line "x1". Exception must be handled. If we handle it, the code compiles without an issue and prints "`Awake!`".