Homework 8

Q1: When try to execute code the compiler gives the error. The error contains the need of **finally** statement to complete block statement.
Finally block is executed even if an unexpected exception don't occurs. Finally block also execute if a try block exits by using a return break or continue. If a catch block throws an exception the finally block still executes.

Answer is option D (True)

Q2: At first the program executes the **try** block if the statements are correct, **finally** block will execute, if **try** block fail then **catch** block will execute and finally execute last.

Answer is option B (True)

Q3: ***Throwable*** is a class that subclass of the ***Object*** which is the super class of class ***Exception.*** Class ***Throwable*** has two two direct subclasses. These are **Exception** and **Error** classes. Option D is the only diagram that fits this situation.

Answer is option D (True)

Q4: In ***Throwable*** type classes, the most afraid type is errors because errors typically are recoverable. When the error occurs there is nothing your program can do.
Answer is option A (True)

Q5: If we declare a variable inside of the try block, we cannot access them of the try block. The solution to get rid of this we can declare the variables outside the try and catch blocks. We cannot reach the score variable thus the code give the error message.

Answer is option D (True)

Q6: A **checked exception** must either caught in a catch block. The other way declared in a throws clause. Such exceptions often indicate serious problems is going through the program termination. *ClassNotFoundException, IOException, NoSuchMethodException* all of this checked exceptions in Java.

An **unchecked** (named with run-time) **exceptions** not caught in catch block or declared in a throw clause. They usually thrown during the evaluation of an expression in predefined classes. To cope with this type of exceptions we need to fix the problem. This uncaught run-time exception terminates program execution. *ArrayIndexOutOfBoundsException, ArithmeticException, NullPointerException* are examples of unchecked exceptions.

Answer is option B (True)

Q7: A keyword ***throw*** is used to throw an exception, whereas **throws** is used in a method's heading to declare an exception. Thus, a throw statement throws an exception, but a throws clause declares one.

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

Answer is option A (True)

Q8: Catching the subclass types separate from superclass. Catching of superclass guarantees that objects of all subclasses will be caught. Position of a catch block for the super class must be last catch block for ensure all subclasses are eventually caught.

Answer is option B (True)

Q9: The code compile correctly without **throw t** expression. It return AC and a stack trace that include RuntimeException but t cannot resolve a type and this create the error.

Answer is option D (True)

Q10: This code does not compiling because the openDrawbridge() method declared the Throws exception expression but main method does not handle the exception. It gives the error in p3 line

Answer is option C (True)

Q11: An **unchecked** (named with run-time) **exceptions** not caught in catch block or declared in a throw clause. They usually thrown during the evaluation of an expression in predefined classes. *ArithmeticException, NullPointerException* are examples of unchecked exceptions. Unchecked exception also known as Runtime Exception. Unchecked exceptions derived from the class RuntimeException and it is not declared. There are only one type exception to handled by the method.

Answer is option B (True)

Q12: The code is compiled correctly. The control flow of code enter the try block and then it break it out with the throwing ClassCastException. The catch block after the the skipped one continues the execution. The other catch black and finally block executed. Therefore we see the 1345 on screen.

Answer is option A (True)

Q13: A finally block must be associated with a try block. We cannot use **finally** keyword without a **try** block. If we place a **finally** block then it will always run after the execution of try block. Thus, option B is false. In normal case when there is no exception in try block then the finally block is executed after try block. An exception in the **finally** block behaves like any other exception. Since option, D is false. Using of the System.exit() command and due to an exception arising in the final block the finally block does not execute.

Answer is option C (True)

Q14: In this code does not compile. When we try to run it tells us its already handled by catch block for IOException. FileNotFoundException is a subclass of IOException. The problem solved with changing the subclass exception with the superclass exception. Moreover, it prints ZY.
Answer is option C (True)

Q15: The optional finally block /clause consist of finally keyword and enclosed in curly braces. Catch keyword must have require the catch block to caught an exception. Finalize method provides the object can eligible to delete the garbage collector. A exception may be not include catch clause but there is should be finally keyword use.

Answer is option C (True)

Q16: In use of exceptions allows the compiler to help the programmer write code that handle of different kind of errors. This situation does not prevent the abnormal terminal program it termination. It tells us there is an problem can be handled a way.

Answer is option B (True)

Q17: The code does not compiled because of catch statement can not enclosed with curly braces. The parenthesis used in given code.

Answer is option D (True)

Q18: A method written is override to allowed a class or not in given options. If the subclass method declares an exception, subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception. If the superclass method does not declare an exception subclass overridden method cannot declare the checked exception but it can declare unchecked exception.

Print exception given in the class The overridden method can include same type exception or not include any exception. The override method don't allow the enhance type exception. Like a Exception

Answer is option B (True)

Q19: Java util type exceptions that belongs the *java.util* package. All of them in package subclass of runtime exception. This exception classes represented in *java.lang.RuntimeException* .This class a subclass of *java.lang.Exception*. Exception classes are subtypes of *java.lang.Exception* class. Top of the exceptions comes from java.lang package. This java lang package imported by default.Usage of this package unnecessary. There is no implementation of java.exception so this option is discarded.

Answer is option D (True)

Q20: Tin exception handling basic syntax consit of **try-catch-finally** block. Code is placed that have possibility of throw exception in **try** blocks. The **catch** part executed only when an exception is thrown it represent the exception type and reference type of variable. **Finally** part always excute if the program throw an exception or not. In this question try catch statements behave like if else structure. The declaration of catch expression is missing. So the code is not compiled in line g3.

Answer is option C (True)

Q21: In some code within a method throws a checked exception then method must either handle with the exception. The other way is it specifies the exception using **throws** keyword. The sentence should be like: A program must handle or declare checked exceptions but should never handle java.lang.Error

Answer is option B (True)

Q22: The code does not compile at line q2 because CastleUnderSiegeException should have declared in line q1. To cover this CastleUnderSiegeException must be declared.

Answer is option B (True)

Q23: If an exception matches two or more catch blocks, the first catch block is executed.

Answer is option A (True)

Q24: This code does not compile because the compile method declared as Exception but catch block includes the NullPointerException. Exception class is does not coverage of NullPointerException and the code does not compile.

Answer is option C (True)

Q25: If the list array assigned with null value the code gives the NullPointer exception.

If the list array have less than ten elements, the code gives the ArrayIndexOutOfBounds exception.

If the boolean array declared as type of object code may give the ClassCastExceptions.

There is only option D is true.

Answer is option D (True)

Q26: A StackOverflow 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.

Answer is option B (True)

Q27: Checked exception as IOException has some methods from the java.io package might throw. Use either a try-catch or adding throws IOException to the method declaration and delegate exception handling to the method caller.

Answer is option C (True)

Q28: This code does not compile because when using the try catch block the order become like try, catch, finally but this code has wrong order.

Answer is option D (True)

Q29: A try statement has optionally finally block. It may be not any. In addition, it has zero or more catch blocks.

Answer is option A (True)

Q30: This code gives the *java.lang.ArithmeticException* by zero because integer type not declared and initialized with default value zero. Java does not allow the division operation with zero.

Answer is option D (True)

Q31: Finally blocks always runs when the code flow break up the try block. There is some except points for instance When you use System.exit(0) or Virtual machine can destroyed or hardware can fault in some way. In question user sees the code flow or exception from finally block

Answer is option B (True)

Q32: This code does not compile because when method declared which is the throws an exception, using a **throws** word not a throw word. This mistyping find in line m1

Answer is option A (True)

Q33: The given code tells about casting operation. Exception class is broader than Runtime Exception and this occurs the ClassCastException.

Answer is option A (True)

Q34: The Throwable classes the broadest class in the options and Java terminology

Answer is option C (True)

Q35: This code does work properly when the give the values in the question. Whether or not this code run the finally statement. So the return expression must include the posted: prefix.

Answer is option D (False) Right option is B

Q36: It depends on a writing order but if we assume the order of question, the catch block for ClassCastException must appear before the catch block for RuntimeException.

Answer is option A (True)

Q37: A method finished with several ways. The code does not compile may with mistyping of semicolon, the computer caught fire this is the worst thing to it. However, the answer is a caller passes invalid data to method.

Answer is option C (True)

Q38: This code does not compile because when overriding a method it can be narrower type exception class can allowed Exception class broader than RuntimeException class

Answer is option C (True)

Q39: This expression may declared with method definition. On other way, this is may be enclosed in try catch block. This line give warnings for the programmer it may be a problem there. The method call can be differ from same exception class type.

Answer is option A (False) Right option is D

Q40: This code does not compile because new kwyword is missing in first try-catch block. And does not produce error message.

Answer is option D (True)

Q41: This code gives the RuntimeException because the most broadest class is RuntimeException

Answer is option C (True)

Q42: This code overrides the catchBall method so it can be covariant of with the original method. This code fits the BadCatchException method. Then to succeeded override it return the same primitive type. Thus the first three option discarded

Answer is option B (False) Right option is D

Q43: This code does not compile because the catch statement has an error type but not have a reference of error type.

Answer is option D (True)

Q44: This class does not compiled and give the stack trace in rumtime

Answer is option D (True)

Q45: The ClassCastException is same type exception of IllegalArgumentException. Two of them is belongs the Runtime exception. Which exception in the first catch statement its evaluated first.

Answer is option C (True)

Q46: This code does not compile because the Biggerproblem class type exceptions not an subclass Throwable class.

Answer is option D (True)

Q47: This code does not compiled because the throws keyword must be changed with throw in try block.

Answer is option D (True)

Q48: Error means in java non-recoverable changes should occurred. When the two users register to account, same time give the warning message (like an exception) to wait them, application lose connection temporarily means the connection will come back later. This is recoverable thing. A user enters their password incorrectly; this will be warned by a message again. But when the application runs out memory, threatening moments waiting for you.

Answer is option D (True)

Q49: The code gives compile error because there is two exception type reference and these reference use the same name. If the reference name will be differen the code will print the Failed to the screen. This error placed in line z1.

Answer is option C (True)

Q50: This code does not compiled because the snore method can throw new exception at the end but cannot declared in method signature. This is pointed by line x1

Answer is option B (True)