Anthony Wittemann

Algorithm Workbench 1-5, Short Answer 1-10 pg 695

1. B 2. B

D D

E

3. //I’m assuming an array of doubles is an acceptable “numeric” array

public double search(double array[], double val){

try{

for(int i = 0; i < array.length; i++){

if(array[i] == val){

return i;

}

}

throw new Exception();

}

catch (Exception e){

System.out.println("Element not found");

}

return -1;

}

or

public double search(double array[], double val) throws Exception{

for(int i = 0; i < array.length; i++){

if(array[i] == val){

return i;

}

}

throw new Exception("Element not found");

}

4. throw new IllegalArgumentException("Argument cannot be negative");

5.

public class NegativeNumberException extends Exception{

private String message;

public NegativeNumberException(){

}

public NegativeNumberException(String mess){

message = mess;

}

public String toString(){

if(message != null){

return message;

}

return "Argument cannot be negative";

}

}

1) It means that the program is aware of an error that was anticipated

2) It means that the program can take action based on an exception was thrown

3) If no throws declaration, the program will go up the stack looking for something that can handle it and then the program halted and the default exception handler displayed the error messages if nothing else if the stack can handle it. Basically there needs to be a catch statement with every try statement that matches the exception thrown.

4) The finally block *always* executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs. But finally is useful for more than just exception handling — it allows the programmer to avoid having cleanup code accidentally bypassed by a return, continue, or break. Putting cleanup code in a finally block is always a good practice, even when no exceptions are anticipated.

5) After the catch block is executed, the program will resume with the code that appears after the entire try / catch construct.

6) Yes, a subclass should be listed before a superclass otherwise an error will occur because the compiler thinks you are handling the same exception more than once.

7) Any throwable objects (includes objects that subclasses of the throwable object)

8) when a throw statement is not surrounded by try,catch

9) Unchecked exceptions:

* represent defects in the program (bugs) - often invalid arguments passed to a non-private method. To quote from The Java Programming Language, by Gosling, Arnold, and Holmes: "Unchecked runtime exceptions represent conditions that, generally speaking, reflect errors in your program's logic and cannot be reasonably recovered from at run time."
* are subclasses of [RuntimeException](http://docs.oracle.com/javase/7/docs/api/java/lang/RuntimeException.html), and are usually implemented using [IllegalArgumentException](http://docs.oracle.com/javase/7/docs/api/java/lang/IllegalArgumentException.html), [NullPointerException](http://docs.oracle.com/javase/7/docs/api/java/lang/NullPointerException.html), or [IllegalStateException](http://docs.oracle.com/javase/7/docs/api/java/lang/IllegalStateException.html)
* a method is not obliged to establish a policy for the unchecked exceptions thrown by its implementation (and they almost always do not do so)

Checked exceptions:

* represent invalid conditions in areas outside the immediate control of the program (invalid user input, database problems, network outages, absent files)
* are subclasses of [Exception](http://docs.oracle.com/javase/7/docs/api/java/lang/Exception.html)
* a method is obliged to establish a policy for all checked exceptions thrown by its implementation (either pass the checked exception further up the stack, or handle it somehow)

10) A throw statement is an action like the return statement. A throws clause is a header that gives information about what a method can do kinda like extends is used for classes.