THROWS Keyword in JAVA

Java keyword: throw and throws. The throw keyword is used to throw an exception from within a method. When a throw statement is encountered and executed, execution of the current method is stopped and returned to the caller. Whereas the throws keyword is used to declare that a method may throw one or some exceptions.

What is a throw in Java?

The **throw** keyword in **Java** is used to explicitly **throw** an exception from a method or any block of code. We can **throw** either checked or unchecked exception. The **throw** keyword is mainly used to **throw** custom exceptions.

What is the use of a throw?

The **throw** blanket is a common accessory to furnishings. They're more than simply functional; they are style makers that add pattern, color and warmth to any room. They are lightweight, can be draped anywhere, and make fantastic housewarming and holiday gifts.

What is the use of throws in Java?

**Throws clause in java** – Exception handling. ... Checked exception (compile time) force you to handle them, if you don't handle them then the program will not compile. On the other hand unchecked exception (Runtime) doesn't get checked during compilation. **Throws** keyword is used for handling checked exceptions.

[**Java keyword: throw and throws**](http://www.codejava.net/java-core/the-java-language/throw-and-throws-keywords)

Last Updated on 19 November 2017   |   [Print](http://www.codejava.net/java-core/the-java-language/throw-and-throws-keywords?tmpl=component&print=1&page=) [Email](http://www.codejava.net/component/mailto/?tmpl=component&template=protostar&link=f8a41e33f85891bb0d11638823a3275fe3f5bff4)

[**Java Spring Framework Masterclass: Beginner to Professional**](https://click.linksynergy.com/link?id=d1Fh24*EiQI&offerid=323058.1132057&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fjava-spring-framework-masterclass%2F&u1=WebTitleSpring)

The **throw** keyword is used to throw an exception from within a method. When a **throw** statement is encountered and executed, execution of the current method is stopped and returned to the caller.

Whereas the **throws** keyword is used to declare that a method may throw one or some exceptions. The caller has to catch the exceptions (catching is optional if the exceptions are of type unchecked exceptions).

These two keywords are usually used together as depicted the following form:

void aMethod() throws Exception1, Exception2 {

// statements...

if (an exception occurs) {

throw new Exception1();

}

// statements...

if (another exception occurs) {

throw new Exception2();

}

}

**Rules**

* 1. *Exception1, Exception2, …*: the exception class must be a direct subclass of Throwable class one of its subclasses.
  2. The **throws** keyword can be followed by one more exception class, separated by commas.
  3. The **throw** keyword must be followed by an instance of Throwable class or one of its subclasses.
  4. When using the **throw** keyword to throw a checked exception from within a method, the method must either:
     + - Declares the **throws** clause followed by the exceptions thrown by the **throw** statements, or:
       - Catches the exceptions thrown by the **throw** statements.
  5. When a method contains statements which may throw exceptions (not using **throw** statements explicitly), it also has to either catch or declare to throw the exceptions.
  6. If the **throw** statements throw unchecked exceptions, the method is not required to declare those unchecked exceptions in its **throws** clause.
  7. A concrete method can declare **throws** clause if only if its body throws checked exceptions. Otherwise a compile error occurs.
  8. An interface’s method can declare **throws** clause freely.
  9. The **throws** clause can declare exceptions which are super types of the exception thrown by the **throw** statements, but not sub types.

**Code Examples**

An interface declares a method that throws an exception:

interface AutoMobile {

void startEngine() throws EngineStartException;

void go();

}

Where EngineStartException is a subclass of Exception class whose super type is Throwable:

class EngineStartException extends Exception {

}

|  |  |
| --- | --- |
| 1  2 | class EngineStartException extends Exception {  } |

A typical usage of **throw** statement and **throws** clause together:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void deleteFile(File file) throws FileNotFoundException {        if (!file.exists()) {          throw new FileNotFoundException();      }      file.delete();  } |

The following example shows a method must declare to throw an exception because it contains the code that may throw an exception:

|  |  |
| --- | --- |
| 1  2  3  4  5 | void writeToFile(String filePath) throws IOException {        BufferedWriter writer = new BufferedWriter(new FileWriter(filePath));      // writes to file...  } |

The method parseInt() in the following code may throw a NumberFormatException which is an unchecked exception, so the method is not required to catch or throw that exception:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | int parseNumber(String input) {        int number;        // this may throw unchecked exception: NumberFormatException      number = Integer.parseInt(input);        return number;  } |

Dice rol

|  |  |
| --- | --- |
|  | import java.util.Scanner;  import java.util.Random;  public class DiceProbability  {public static void main(String[] args)      {          Scanner in = new Scanner(System.in);          Random generator = new Random();            int ones = 0;          int twos = 0;          int threes = 0;          int fours = 0;          int fives = 0;          int sixes = 0;          int sevens = 0;          int eights = 0;          int nines = 0;          int tens = 0;          int elevens = 0;          int twelves = 0;              System.out.println("Please enter a number of dice rolls");          int rolls = in.nextInt();            for(int start = 0; start < rolls; start++)          {int random = generator.nextInt(12) + 1;              for(int side1= 0; start < rolls;)              {if(random == 1)                  {ones++;}                  else if(random == 2)                  {twos++;}                 else if(random == 3)                  {threes++;}                 else if(random == 4)                  {fours++;}                 else if(random == 5)                  {fives++;}                 else if(random == 6)                  {sixes++;}                 else if(random == 7)                  {sevens++;}                 else if(random == 8)                  {eights++;}                 else if(random == 9)                  {nines++;}                 else if(random == 10)                  {tens++;}                 else if(random == 11)                  {elevens++;}                 else if(random == 12)                  {twelves++;}              }          }            System.out.println("Number of rolls: " + rolls);          System.out.println("Sum of dice: " + "              " + "Probability");          System.out.println("1's        : " + "              " + (double)ones/(double)rolls \* 100 + "%");          System.out.println("2's        : " + "              " + (double)twos/(double)rolls \* 100 + "%");          System.out.println("3's        : " + "              " + (double)threes/(double)rolls \* 100 + "%");          System.out.println("4's        : " + "              " + (double)fours/(double)rolls \* 100 + "%");          System.out.println("5's        : " + "              " + (double)fives/(double)rolls \* 100 + "%");          System.out.println("6's        : " + "              " + (double)sixes/(double)rolls \* 100 + "%");          System.out.println("7's        : " + "              " + (double)sevens/(double)rolls \* 100 + "%");          System.out.println("8's        : " + "              " + (double)eights/(double)rolls \* 100 + "%");          System.out.println("9's        : " + "              " + (double)nines/(double)rolls \* 100 + "%");          System.out.println("10's       : " + "              " + (double)tens/(double)rolls \* 100 + "%");          System.out.println("11's       : " + "              " + (double)elevens/(double)rolls \* 100 + "%");          System.out.println("12's       : " + "              " + (double)twelves/(double)rolls \* 100 + "%");        }  } |

import java.util.Scanner;

import java.util.Random;

public class DiceProbability

{public static void main(String[] args)

    {

        Scanner in = new Scanner(System.in);

        Random generator = new Random();

        int[] a = new int[12];

        System.out.println("Please enter a number of dice rolls");

        int rolls = in.nextInt();

        for(int start = 0; start < rolls; start++)

        {int random = generator.nextInt(12) + 1;

            for(int side1= 0; start < rolls;)

            {a[random]++;

            }

        }

        System.out.println("Number of rolls: " + rolls);

        System.out.println("Sum of dice: " + "              " + "Probability");

        System.out.println("1's        : " + "              " + (double)a[1]/(double)rolls \* 100 + "%");

        System.out.println("2's        : " + "              " + (double)a[2]/(double)rolls \* 100 + "%");

        System.out.println("3's        : " + "              " + (double)a[3]/(double)rolls \* 100 + "%");

        System.out.println("4's        : " + "              " + (double)a[4]/(double)rolls \* 100 + "%");

        System.out.println("5's        : " + "              " + (double)a[5]/(double)rolls \* 100 + "%");

        System.out.println("6's        : " + "              " + (double)a[6]/(double)rolls \* 100 + "%");

        System.out.println("7's        : " + "              " + (double)a[7]/(double)rolls \* 100 + "%");

        System.out.println("8's        : " + "              " + (double)a[8]/(double)rolls \* 100 + "%");

        System.out.println("9's        : " + "              " + (double)a[9]/(double)rolls \* 100 + "%");

        System.out.println("10's       : " + "              " + (double)a[10]/(double)rolls \* 100 + "%");

        System.out.println("11's       : " + "              " + (double)a[11]/(double)rolls \* 100 + "%");

        System.out.println("12's       : " + "              " + (double)a[12]/(double)rolls \* 100 + "%");

}}

package edu.uba.filters;

import com.google.common.collect.LinkedListMultimap;

import com.google.common.collect.Multimap;

import com.google.common.collect.Multiset;

import com.google.common.collect.HashMultiset;

import java.util.Collections;

import java.util.Comparator;

import java.util.Iterator;

import java.util.Set;

public class Frequency<T extends Comparable<T>> {

private Multiset event = HashMultiset.create();

private Multimap event2 = LinkedListMultimap.create();

public void addValue(T data){

if(event2.containsKey(data) == false){

event2.put(data,data);

}

event.add(data);

}

public void clear(){

this.event = null;

this.event2 = null;

this.event = HashMultiset.create();

this.event2 = LinkedListMultimap.create();

}

public double getPct(T data){

int numberOfIndElements = event.count(data);

int totalNumOfElements = event.size();

return (double) numberOfIndElements/totalNumOfElements;

}

public int getNum(T data){

int numberOfIndElements = event.count(data);

return numberOfIndElements;

}

public int getSumFreq(){

return event.size();

}

public int getUniqueCount(){

return event.entrySet().size();

}

public String[] getKeys(){

Set<String> test = event2.keySet();

Object[] keys = test.toArray();

String[] keysAsStrings = new String[keys.length];

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

keysAsStrings[i] = (String) keys[i];

}

return keysAsStrings;

}

}