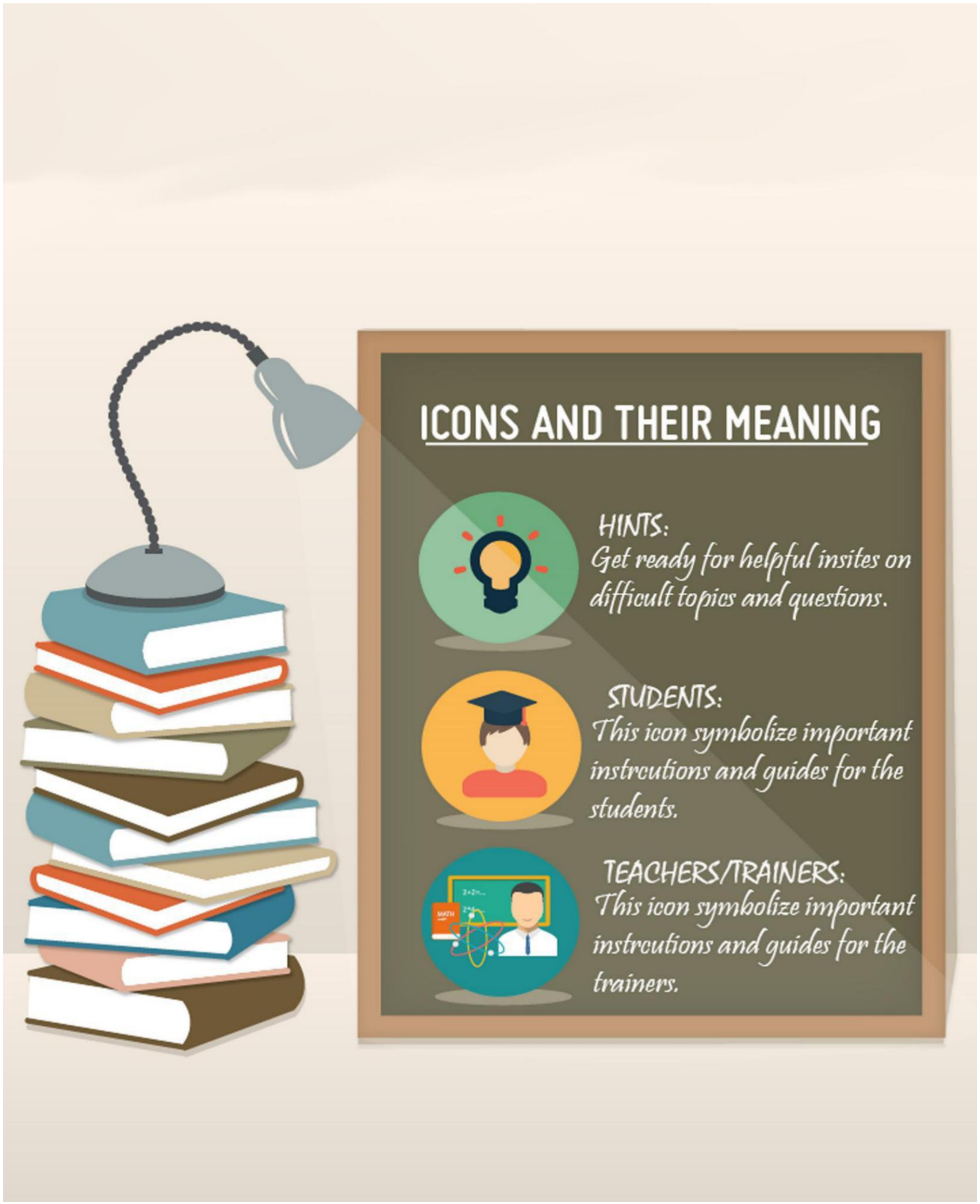
CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 6: Thread and Exception Handling

Chapter 5

Objective: After completing this lesson you will be able to : * Gain an understanding of the throw and throws keywords in Java * Understand the try-catch-finally block in Java * Gain an introduction into custom exceptions in Java * Learn about exception control flow	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 60 minutes	**Practical Duration:** 60 minutes
Total Duration: 120 minutes	

Chapter 5

5.1 Throw vs. Throws

Throw and throws are two keywords used in Java for throwing exceptions within programs. While the syntax is somewhat similar, there are differences between throw and throws.

Throw and throws differences

The throw and throws keywords can be differentiated based on –

i) Usage

Throw – A throw keyword is used to throw an exception logically, within a function.

Throws – Throws is used within the signature of a function. It is used when there is a probability of some statements resulting in exceptions.

ii) Number of exceptions

Throw – A throw keyword can throw one exception at a time.

Throws – Throws can be used for declaring multiple exceptions, by separating them with the use of commas. When one of these declared exceptions occur, it is thrown automatically.

iii) Exception propagation

Throw – A throw keyword can be used for propagating only unchecked exceptions. It is incapable of checked exception propagation.

Throws – The throws keywords can be used for propagating only checked exceptions. It is incapable of unchecked exception propagation.

iv) Syntax

Throw – The throw keyword contains the exception instance.

Throws – The throws keyword contains the exception class names.

*** Code example of the throw keyword in Java**

```
public class ThrowDemo1{
  static void validate(int age){
   if(age<24)
   throw new ArithmeticException(‘not valid‘);
   else
     System.out.println(‘welcome to the club‘);
  }
  public static void main(String args[]){
     validate(17);
     System.out.println(‘code remainder‘);
  }
}
```

Output:

```
Exception in thread main java.lang.ArithmeticException:not valid
```

*** Code example of the throws keyword in Java**

```
import java.io.IOException;
Class Throwsdemo1{
  void q()throws IOException{
   throw new IOException(‘error‘);
}
  void r()throws IOException{
    q();
  }
  void s(){
  try{

    r();
  }catch(Exception e){System.out.println('exception has been handled');}
  }
  public static void main(String args[]){
  Throwsdemo1 obj=new Throwsdemo1();
  obj.s();
  System.out.println('flow is normal');
  }
}
```

Output:

```
exception has been handled
flow is normal
```

5.2 Try-Catch-Finally Block

In Java, the try, catch and finally blocks are useful tools for writing code for applications. These enable the throwing of runtime exceptions. Using these blocks let programmers process an exception by -

* execution a different application logic

* handling the said exception

The use of try, catch and finally blocks helps to prevent application crash events.

i) **try block** - A try block contains application code meant to work under conventional circumstances. It performs tasks like file reading and writing to databases.

Syntax -

```
try {
   //code body
}
```

ii) **catch block** – Catch blocks are optional and follow the try blocks. They have to handle the checked exceptions that are thrown by try blocks. Catch blocks can also handle unchecked exceptions. Multiple catch blocks can be associated with one try block.

Syntax –

```
try {
   //program code
}
catch(Exception e) {
   //exception handling
}
```

iii) **finally block** – It is another optional block suitable for running code that is meant to be executed every time, regardless of the presence of errors.

Syntax –

```
try {
   //file open
   //reading file
}
catch(Exception e) {
   //exception handling during file reading
}
finally {
   //file close
}
```

5.3 Custom Exception

In Java, a custom exception or user-defined exception is one that is created by a programmer. A custom exception is utilized to modify an exception according to the needs of users. Programmers can include customized messages with custom exceptions.

Example of custom exception –

```
class MyException extends Exception {
  int id;


  public MyException(int x) {


    id = y;
  }


  public String toString() {
    return ‘CustomException[‘ + id + ‘]‘;
  }
}


public class Samples {
  static void compute(int b) throws MyException {
    if (b | 10)
      throw new MyException(b);
      System.out.println(‘No prog error. no exception present‘);
  }


  public static void main(String args[]) {


    try {
      compute(6);
      compute(13);
    } catch(MyException ex1) {
      System.out.println(ex1);
    }
  }
}
```

Output:

No prog error. no exception present

CustomException[13]

5.4 Control flow in exception

Java exceptions can be used for flow control purposes within programs. An exception can transfer the flow of control from a location where a problem is detected to one where it can be dealt with. An exception is detected and thrown from one place and caught with an exception handler in a different program segment.

To gain a better understanding of control flow in exception, take a look at the example code involving the try-catch-finally clause –

```
class Exceptional
{
   public static void main (String[] args)
   {
      // array size is 5.
      int[] arr = new int[5];
      try
      {
         int i = arr[5];

         // the statement cannot ever be executed as an exception is initiated by the statement
         System.out.println('within try block');
      }
      catch(ArrayIndexOutOfBoundsException ex)
{
         System.out.println('catch block has an exception');
      }
      finally
      {
         System.out.println('finally block is executed');
      }
      // remainder of the program to be executed
      System.out.println('Outside of the try-catch-finally clause');
```

```
 }
}
```

Output:

```
Catch block has exception
finally block is executed

Outside of the try-catch-finally clause
```

Practical (60 **minutes**)

See the example programme for Java throw keyword below. Write the same programme to throw the exception string value "invalid" if integer value of age is less than 22, with else condition "welcome aboard" . Validate for age value 19. Show the resulting output. Repeat the same programme to validate for age value 24.

```
public class ThrowDemo1{

  static void validate(int age){

   if(age<24)

   throw new ArithmeticException('not valid');

   else

     System.out.println('welcome to the club');

 }

 public static void main(String args[]){

     validate(17);

     System.out.println('code remainder');

 }

}
```

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ (10 minutes)

1. Throw and __________ are two Java keywords used for throwing exceptions.

a) throws

b) throwing

c) thrown

d) None of the mentioned

2. What keyword is used for logically throwing an exception within a function?

a) throw

b) throws

c) throwing

d) None of the mentioned

3. Throws is used within the __________ of a function.

a) encapsulation

b) body

c) signature

d) None of the mentioned

4. How many errors can throw exception throw?

a) one

b) multiple

c) two

d) None of the mentioned

5. What are used by throws to separate multiple exception declarations?

a) semi-colons

b) commas

c) colons

d) None of the mentioned

6. What keyword can propagate only unchecked exceptions?

a) throw

b) throws

c) throwing

d) None of the mentioned

7. The throw keyword contains the exception ___________.

a) instants

b) instance

c) enumerations

d) None of the mentioned

8. A catch block is used for enabling the throwing of ________ exceptions.

a) catcher

b) catching

c) runtime

d) None of the mentioned

9. Which block runs regardless of the presence of errors?

a) try

b) catch

c) finally

d) None of the mentioned

10. Java exceptions can be utilized for _______ control within programs.

a) flow

b) flowing

c) flowed

d) None of the mentioned