জাভায় এক্সসেপশন হ্যান্ডলিং কৌশল

মো: আবদুল কাদের

জাভায় এরর বা সমস্যা সমাধানের জন্য এক্সসেপশন হ্যান্ডলিং (Exception handling) ব্যবস্থা অত্যন্ত শক্তিশালী। এটি রান টাইমে কোনো সমস্যা দেখা দিলেও জাভা প্রোগ্রামকে সঠিকভাবে চলতে সহায়তা করে। এক্সসেপশনের অর্থ অস্বাভাবিক অবস্থা বা ব্যতিক্রম অবস্থা, যা প্রোগ্রামের স্বাভাবিক ফ্লোকে বাধাগ্রস্ত করে। ধরা যাক, প্রোগ্রামে একটি কোডের মাধ্যমে আমরা ডাটাবেজের সাথে সংযোগ স্থাপন করতে চাই, কিন্তু আদতে সেই ডাটাবেজটি নেই বা তৈরি করাই হয়নি অথবা ডাটাবেজটি অন্য নামে রয়েছে। তাহলে প্রোগ্রাম ওই ডাটাবেজের সাথে সংযোগ স্থাপন করতে পারবে না। আবার যদি সংখ্যা নিয়ে কাজ করতে চাই, তাহলে সেটাকে প্রথমে সংখ্যায় পরিবর্তন করে নিতে হবে। বাই ডিফল্ট রান টাইমে গ্রহণ করা সংখ্যা স্ট্রিং ডাটা হিসেবে থাকে। ফলে সংখ্যাবাচক ডাটার মতো ব্যবহার করতে চাইলে একটি ইভেন্ট সংঘটিত হবে, যাকে জাভায় বলা হয় এক্সসেপশন। এটি প্রোগ্রাম কোডজনিত কোনো ভুল নয়। প্রোগ্রামের কোডজনিত ভুল থাকলে কম্পাইল করাই যাবে না। বিভিন্ন ধরনের এক্সসেপশন নিয়ে কাজ করার জন্য জাভায় বিভিন্ন এক্সসেপশন ক্লাস রয়েছে। যেমন– ArithmeticException, NumberFormatException, IOException ArrayIndexOutOfBoundsException, ClassNotFoundException SQLException ইত্যাদি।

চেকড ও আনচেকড নামে দুই ধরনের এক্সসেপশন রয়েছে। RuntimeException ছাড়া যেসব ক্লাস Throwable ক্লাসকে এক্সটেন্ড করে, তাদেরকে চেকড এক্সসেপশন বলে। যেমন– IOException, SQLException ইত্যাদি। এসব এক্সসেপশন কম্পাইল করার সময় চেক করা হয়।

যেসব ক্লাস RuntimeException ক্লাসকে এক্সটেন্ড করে, তাদেরকে আনচেকড এক্সসেপশন বলে। যেমন– ArithmeticException, NullPointerException,ArrayIndexOutOfBoundsException ইত্যাদি। এসব এক্সসেপশন রান টাইমে চেক করা হয়।

জাভায় বিভিন্ন ধরনের এক্সসেপশনের উদাহরণ

ক. **Arithmetic Exception**

```
int a=50/0;
```

খ. **Number Format Exception**

```
String s=”abc”;
int i=Integer.parseInt(s);
```

গ. **Array Index Out Of Bounds Exception**

```
int a[]=new int[5];
a[10]=50;
```

এক্সসেপশন হ্যান্ডলিং কিওয়ার্ড

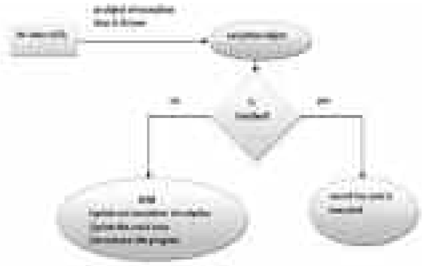
try, catch, finally, throw এবং throws এই পাঁচটি কিওয়ার্ডের মাধ্যমে জাভায় এক্সসেপশন হ্যান্ডলিং করা হয়।

try-catch block

```
try{
//code that may throw exception
}catch(Exception_class_Name ref){}
```

try-finally block

```
try{
//code that may throw exception
}finally{}
```



try-catch এবং try-finally-এর মধ্যে পার্থক্য হলো try-এ লেখা কোডে কোনো এক্সসেপশন হলে তা catch-এ উল্লিখিত এক্সসেপশন ক্লাসের কাছে পাঠাবে। আর try-finally-এর মাধ্যমে try ব্লকে কোনো এক্সসেপশন ঘটলে তা কোনো এক্সসেপশন ক্লাসের কাছে পাঠাতে বা নাও পাঠাতে পারে, তবে চূড়ান্তভাবে একটি কাজ করবে, যা finally ব্লকের মধ্যে দেয়া থাকবে।

```
public class Testtrycatch1
{
public static void main(String args[])
{
Try
{
int data=50/0;
}
catch(ArithmeticException e)
{
System.out.println(e);
}
System.out.println(“finally executed”);
}
}
```

একটি try ব্লকের জন্য কয়েকটি catch ব্লক থাকতে পারে। যদি কয়েকটি এক্সসেপশন তৈরি হয়, সে ক্ষেত্রে প্রত্যেকটি এক্সসেপশন আলাদাভাবে সমাধান করার জন্য আলাদা catch ব্লক ব্যবহার করা হয়। তবে সে ক্ষেত্রে সিনিয়রিটি মেনে ব্যবহার করতে হবে। যেমন–

```
try
{ Sample Code }
catch (IndexOutOfBoundsException e) {
System.err.println(“IndexOutOfBoundsException: “ + e.getMessage());
} catch (IOException e) {
System.err.println(“Caught IOException: “ + e.getMessage());
}
```

try-finally-এর উদাহরণ

```
class TestFinallyBlock{
public static void main(String args[]){
try{
int data=25/5;
System.out.println(data);
}
catch(NullPointerException  e){System.out.println(e);}finally{System.out.println(“finally block is always executed”);}
System.out.println(“rest of the code...”);
}
}
```

finally ব্লক ততক্ষণ পর্যন্ত এক্সিকিউট হবে না, যতক্ষণ না পর্যন্ত প্রোগ্রাম বন্ধ হয়ে যাচ্ছে অথবা এমন কোনো ঘটনা ঘটবে যাতে প্রোগ্রাটি না রান করে বন্ধ হয়ে যাবে কজ

ফিডব্যাক : ***balaith@gmail.com***