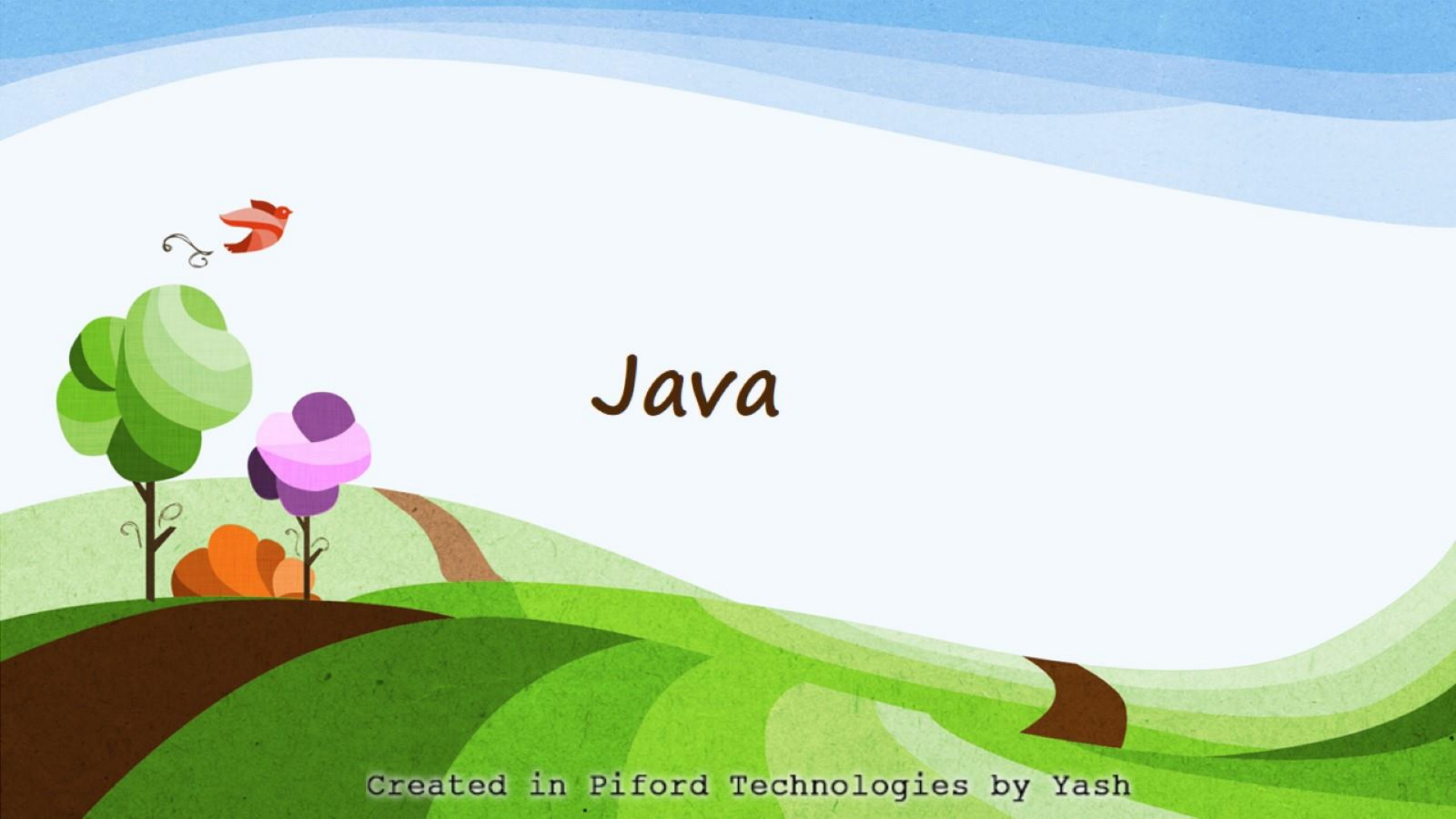
Java
Created in Piford Technologies by Yash

Exception Handling

Created in Piford Technologies by Yash

Exception Handling

- The exception handling in java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

What is exception handling

- Exception Handling is a mechanism to handle runtime errors such as Class Not Found, IO, File Not Found etc.

Types of Exception

- There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions:
  - Checked Exception
  - Unchecked Exception
  - Error

- **1) Checked Exception**

The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

- **2) Unchecked Exception**

The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

- **3) Error**

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

Java Exception Handling Keywords

1. Try
2. Catch
3. Finally
4. Throw
5. Throws

Try-Catch

- Java try block is used to enclose the code that might throw an exception. It must be used within the method.

Example without Exception:

*Hornet.java

```
1  package try5;
2
3  class Hornet {
4      public static void main(String[] args) {
5          int d= 9/0;
6          System.out.println(d);
7      }
8  }
9
```

```
<terminated> Hornet [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at try5.Hornet.main(Hornet.java:6)
```

Example with Exception:

Hornet.java

```
1  package try5;
2
3  class Hornet {
4      public static void main(String[] args) {
5          try{
6                  int d= 9/0;
7                  System.out.println(d);
8          }catch(ArithmeticException m){
9              System.out.println("Something Went Wrong");
10         }
11         System.out.println("Hello World");
12     }
13 }
```

```
Something Went Wrong
Hello World
```

Multi catch block

- If you have to perform different tasks at the occurrence of different Exceptions, use java multi catch block.

Catch multiple exceptions

Hornet.java

```
1  package try5;
2
3  class Hornet {
4      public static void main(String[] args) {
5          try{
6                  int a[]= new int[9];
7                  a[10]=976;
8          }catch(ArithmeticException m){
9              System.out.println("Something Went Wrong");
10         }catch(ArrayIndexOutOfBoundsException m2){
11             System.out.println("Something Went Wrong Again");
12         }
13         System.out.println("Hello World");
14     }
15 }
16
```

```
Something Went Wrong Again
Hello World
```

Finally block

- Java finally block is a block that is used to execute important code such as closing connection, stream etc.
- Java finally block is always executed whether exception is handled or not.
- Java finally block follows try or catch block.
- If you don't handle exception, before terminating the program, JVM executes finally block(if any).

Example:

Hornet.java

```
1  package try5;
2
3  class Hornet {
4      public static void main(String[] args) {
5          try{
6                int a[]= new int[9];
7                a[10]=976;
8          }catch(ArithmeticException m){
9              System.out.println("Something Went Wrong");
10         }catch(ArrayIndexOutOfBoundsException m2){
11             System.out.println("Something Went Wrong Again");
12         }finally{
13             System.out.println("Sorry, Website Not Working Properly");
14         }
15         System.out.println("Hello World");
16     }
17 }
18
```

```
Something Went Wrong Again
Sorry, Website Not Working Properly
Hello World
```

Throw Exception

- 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.

Example:

Hornet.java

```
1  package try5;
2
3  class Hornet {
4        static void show(int age){
5            if(age < 18)
6                throw new ArithmeticException("You Are Not Eligible");
7            else
8                System.out.println("You are Ready For Party");
9        }
10     public static void main(String[] args) {
11         show(16);
12     }
13 }
```

```
Exception in thread "main" java.lang.ArithmeticException: You Are Not Eligible
        at try5.Hornet.show(Hornet.java:6)
        at try5.Hornet.main(Hornet.java:11)
```

Throws

- 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).

Example:

```
void deleteFile(File file) throws FileNotFoundException {
    if (!file.exists()) {
        throw new FileNotFoundException();
    }
    file.delete();
}
```