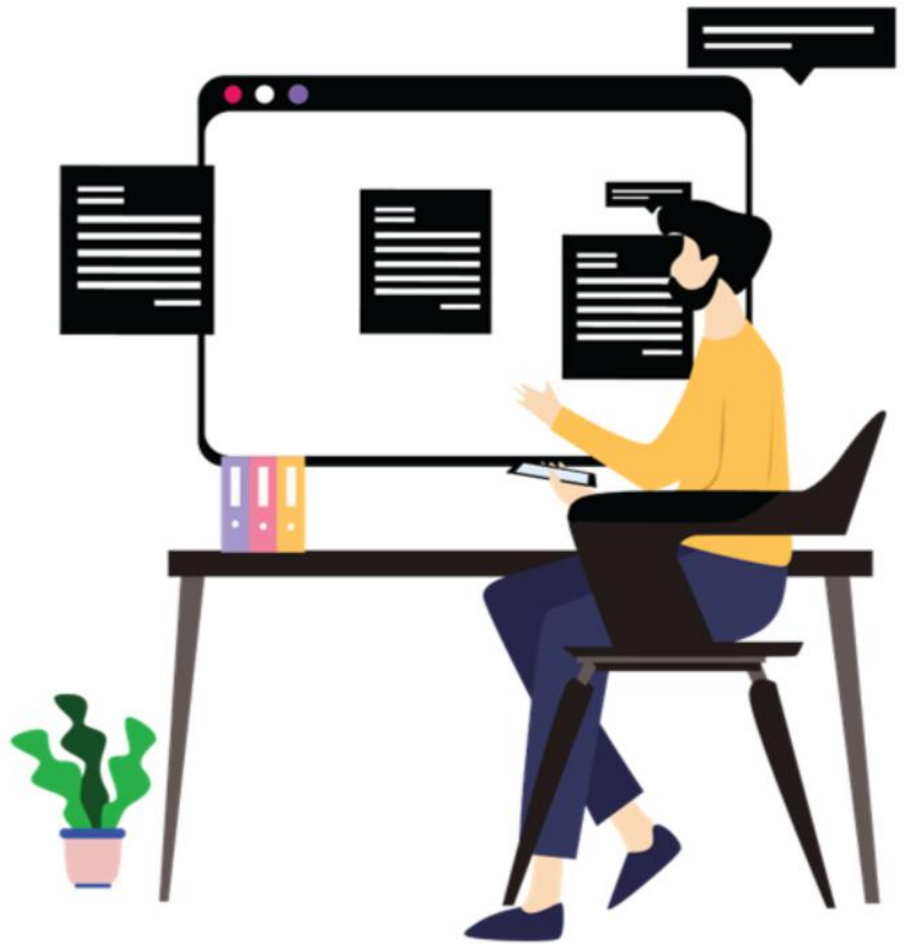
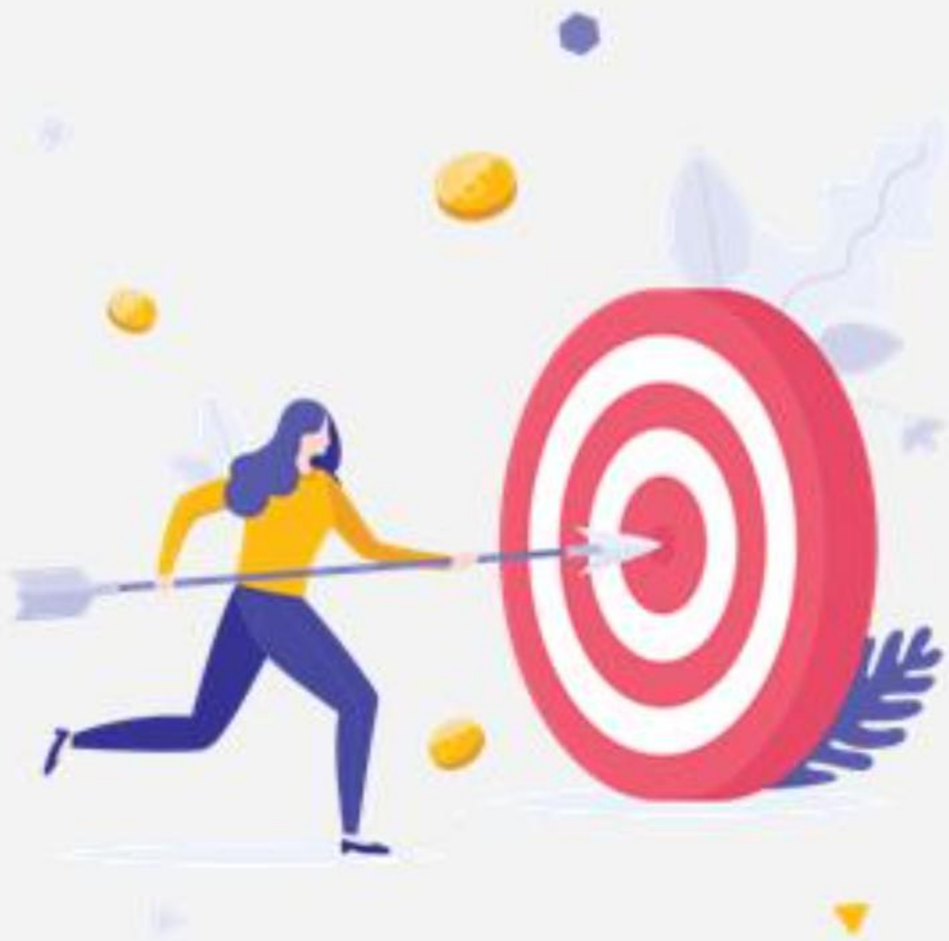
Learning Consolidation

Create and Implement a User-Defined Exception





Learning Objectives

- Understand Checked Exception
- Introduction to User-defined Exception
- Implement Throw and Throws Keyword
- Difference Between Checked and Unchecked Exception

Slide Note

User-Defined Exceptions

- Java allows programmers to create their own exceptions.
    - Such exceptions are known as "custom exceptions" or "user-defined exceptions"
- User-defined exceptions are classes that provide flexibility to add attributes and methods that are not part of a standard Java exception library.
- All user-defined exceptions are **checked exceptions.**

Slide Note

Creating User-Defined Exceptions

- To create a user-defined exceptions:
  - Create an exception class
  - The created class needs to inherit the `java.lang.Exception` class, as shown in the image.
- The `PlayerNotFoundException` is a user-defined exception class.

```java
public class PlayerNotFoundException extends Exception{

    public PlayerNotFoundException(String message) {
        super(message);
    }
}
```

Slide Note

Handling User-Defined Exceptions In a Program

```
public static void getAllPlayers(int age, Player player) throws PlayerNotFoundException {

    //if player of specific age is not found throw the exception
    if(player.getAge()<40){
        throw new PlayerNotFoundException("Player of age above 40 is not Found");
    }
}
public static void main(String[] args) {
    try {
        getAllPlayers( age: 41,player);
    } catch (PlayerNotFoundException e) {
        e.printStackTrace();
    }
}
```

- The `getAllPlayers` method is called from the main method.
- Here, the method `getAllPlayer` is throwing a checked exception using the throws keyword.
- The main method must handle the exception using a try catch block or
- It can propagate it to any other method.
- So, when throwing a checked exception, it must be handled by the calling method using a try. Catch or by declaring using the throws keyword.

Slide Note

Checked vs Unchecked Exceptions

Checked Exceptions	Unchecked Exceptions
They occur at compile time	They occur at runtime
The compiler gives a compile-time error	The compiler does not give a compile-time error
These types of exceptions need to be handled at compile time	These types of exceptions cannot be handled at the time of compilation, because they are generated by logical mistakes in the program
The JVM requires that the exception be caught and handled	The JVM does not require the exception to be caught and handled

Example of Checked Exceptions:
`IOException`
`FileNotFoundException`
`ClassNotFoundException`
	Examples of Unchecked Exceptions:
	`NullPointerException`
	`ArrayIndexOutOfBoundException`
	`ArithmeticException`

Slide Note

Throw vs. Throws

Throw	Throws
The throw keyword is used inside a function or inside a block of code to throw an exception logically.	The throws keyword is used in the method signature to declare an exception that might be thrown by the function while running the code.
The throw keyword throws the exception explicitly. It can only throw one exception object at a time.	Throws can be used to declare multiple exceptions, separated by a comma.

Slide Note

The `try-with-resources` statement

- The `try-with-resources` statement is a try statement that declares one or more resources.
- A *resource* is an object that must be closed after the program is finished with it.
- The try-with-resources statement ensures that each resource is closed at the end of the statement.
- Try with resources are mostly used with File to close the file resources.