Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 3

Lecture 24

Lecture 24

- ForEach Method
- Try-with resources
- Type Annotations, Repeating Annotations

forEach loop

- Java provides a new method forEach() to iterate the elements.
- It is defined in Iterable and Stream interface.
- It is a default method defined in the Iterable interface.
- Collection classes which extends Iterable interface can use forEach loop to iterate elements.

forEach() Signature in Iterable Interface

default void forEach(Consumer<**super** T>action)

Java 8 forEach() example

```
import java.util.ArrayList;
import java.util.List;
public class ForEachExample {
    public static void main(String[] args) {
        List<String> gamesList = new ArrayList<String>();
        gamesList.add("Football");
        gamesList.add("Cricket");
        gamesList.add("Chess");
        gamesList.add("Hocky");
        gamesList.forEach(System.out::println);
    } }
```

Java Stream forEachOrdered() Method Example

```java
import java.util.ArrayList;
import java.util.List;
public class ForEachOrderedExample {
  public static void main(String[] args) {
    List<String> gamesList = new ArrayList<String>();
    gamesList.add("Football");
    gamesList.add("Cricket");
    gamesList.add("Chess");
    gamesList.add("Hocky");
  System.out.println("------Iterating by passing lambda expression--------------");
    gamesList.stream().forEachOrdered(games -> System.out.println(games));
  System.out.println("------Iterating by passing method reference--------------");
    gamesList.stream().forEachOrdered(System.out::println);
  } }
```

Try-with-resources

- Try-with-resources statement is a try statement that declares one or more resources in it.
- A resource is an object that must be closed once your program is done using it.
- For example, a File resource or a Socket connection resource.
- The try-with-resources statement ensures that each resource is closed at the end of the statement execution.
- If we don’t close the resources, it may constitute a resource leak and also the program could exhaust the resources available to it.

- We can pass any object as a resource that implements java.lang.AutoCloseable, which includes all objects which implement java.io.Closeable.
- We don’t need to add an extra finally block for just passing the closing statements of the resources.
- The resources will be closed as soon as the try-catch block is executed.

Java Type Annotations

- Java 8 has included two new features repeating and type annotations in its prior annotations topic.
- In early Java versions, you can apply annotations only to declarations.
- After releasing of Java SE 8 , annotations can be applied to any type use.
- It means that annotations can be used anywhere.

For example, if you want to avoid NullPointerException in your code, you can declare a string variable like this:

@NonNull String str;

Following are the examples of type annotations:

1. @NonNull List<String>
2. List<@NonNull String> str
3. Arrays<@NonNegative Integer> sort
4. @Encrypted File file
5. @Open Connection connection
6. void divideInteger(int a, int b) throws @ZeroDivisor ArithmeticException

Repeating Annotations

- Java allows you to repeating annotations in your source code.
- It is helpful when you want to reuse annotation for the same class. You can repeat an annotation anywhere that you would use a standard annotation.
- For compatibility reasons, repeating annotations are stored in a container annotation that is automatically generated by the Java compiler

In order for the compiler to do this, two declarations are required in your code.

- Declare a repeatable annotation type
- Declare the containing annotation type

1) Declare a repeatable annotation type

Declaring of repeatable annotation type must be marked with the @Repeatable meta-annotation.

In the following example, we have defined a custom @Game repeatable annotation type.

```
@Repeatable(Games.class)
@interfaceGame{
    String name();
    String day();
}
```

2) Declare the containing annotation type

Containing annotation type must have a value element with an array type.

The component type of the array type must be the repeatable annotation type.

In the following example, we are declaring Games containing annotation type:

```
@interfaceGames{
   Game[] value();
}
```

Repeating Annotations Example

```
import java.lang.annotation.Repeatable;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
// Declaring repeatable annotation type
@Repeatable(Games.class)
@Retention(RetentionPolicy.RUNTIME)
@interface Game {
    String name();
    String day();
}
// Declaring container for repeatable annotation type
@Retention(RetentionPolicy.RUNTIME)
@interface Games {
    Game[] value();
}
```

```
// Repeating annotation
@Game(name = "Cricket",  day = "Sunday")
@Game(name = "Hockey",   day = "Friday")
@Game(name = "Football", day = "Saturday")
public class RepeatingAnnotationsExample {
    public static void main(String[] args) {
        // Getting annotation by type into an array
       Game[] games =
RepeatingAnnotationsExample.class.getAnnotationsByType(Game.class);
        for (Game game : games) {    // Iterating values
            System.out.println(game.name() + " on " +
game.day());
        }
    }
}
```

Output

```
Cricket on Sunday
Hockey on Friday
Football on Saturday
```