Methods to print exception message

1. printStackTrace()

Prints complete stack trace of the exception(class, message, line number)

```
package methodToPrintExc;

public class PrintStackTraceExample {
    public static void main(String[] args) {
        try {
           int result = 10/0; //ArithmeticException
        } catch (ArithmeticException e){
            e.printStackTrace();
        }

    }
}
```

o/p:

java.lang.ArithmeticException: / by zero

at methodToPrintExc.PrintStackTraceExample.main(PrintStackTraceExample.java:6)

2. getMessage()

```
package methodToPrintExc;

public class GetMessageExample {
    public static void main(String[] args) {
        try {
            String str = null;
            System.out.println(str.length());  // null pointer exception
        } catch (NullPointerException e) {
            System.out.println("GetMessage: "+ e.getMessage());
        }
```

```
    }
}
```

o/p:

GetMessage: Cannot invoke "String.length()" because "str" is null

3.toString()

Returns a short description(exc. Class name and message)

```
package methodToPrintExc;

public class ToStringExample {
    public static void main(String[] args) {
        try {
            int[] arr = new int[3];
            System.out.println(arr[5]); // AIOBException
        }catch (ArrayIndexOutOfBoundsException e){
            System.out.println("Message: "+ e.toString());
        }
    }
}
```

o/p:

Message: java.lang.ArrayIndexOutOfBoundsException: Index 5 out of bounds for length 3

Custom Exception:

Steps to create custom exceptions:

1. Create a class and extend it accordingly
2. Add a constructor
3. Throw and handle the exception

```
package customExceptionExample;

public class InsufficientFundsException extends
Exception{

    //Constructor with a custom message
    public InsufficientFundsException(String message) {
        super(message);   //Pass this message to the
parent exception class
    }
}
```

```
package customExceptionExample;

public class CustomExceptionExample {

    private double balance;

    public CustomExceptionExample(double balance) {
        this.balance = balance;
    }

    //Method to withdraw the money
    public void withdraw(double amount) throws
InsufficientFundsException{
        if(amount>balance) {
            throw new InsufficientFundsException("Insufficient
Balance!!, Available balance: " + balance);
        }
        balance-=amount;
        System.out.println("Withdraw successful!!!, Remaining
balance: "+ balance);
```

```
    }

    public static void main(String[] args) {
        CustomExceptionExample account = new
CustomExceptionExample(500.0);
        try {
            account.withdraw(200.0);  //transaction is
successful
            account.withdraw(400.0);
        }catch (InsufficientFundsException e){
            System.out.println("EXCEPTION: "+e.getMessage());
        }
        System.out.println("Transaction Successful!!!!");
    }
}
```

o/p:
Withdraw successful!!!, Remaining balance: 300.0
EXCEPTION: Insufficient Balance!!, Available balance: 300.0
Transaction Successful!!!!

- Getters & Setters:
  Encapsulation: by making our variables private, we protect them from direct access.
  Getter: Method which returns the value of a variable
  Setter: Method which sets/update the value

  TASK:
  Create a Product class with:
  Private fields: id,name, price
  Getters and setters
  Validations: price should not be less than zero.

Collection Framework Overview:

- It provides set of interfaces and classes to manage the group of objects.
- It has ready made implementations of data structures.

Framework:

A predefined structure which provides reusable designs

Iterable Interface: Iterable (Java SE 21 & JDK 21)

Root interface which allows the traversing through elements using iterators

Collection Interface: Extends the iterable. It defines the common methods like adding,removing and checking the size of collection.

List Implementations:*