Practical Assignment 4

1.

```
import java.util.Scanner;

class InvalidProductException extends Exception {
    InvalidProductException(String message) {
        super(message);
    }
}

class Product {
    int productCode;
    String productName;
    float weight;

    Product(int code, String name, float weight) {
        this.productCode = code;
        this.productName = name;
        this.weight = weight;
    }

    void display() {
        System.out.println("Product Code: " + productCode);
        System.out.println("Product Name: " + productName);
        System.out.println("Weight: " + weight + "grams");
    }
}

public class Q1 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int code;
        String name;
        float weight;
```

```
        System.out.print("Enter Product Code: ");
        code = sc.nextInt();
        sc.nextLine();
        System.out.print("Enter Product Name: ");
        name = sc.nextLine();
        System.out.print("Enter Product Weight(in grams): ");
        weight = sc.nextFloat();

        Product p = new Product(code, name, weight);

        try {
            if (p.weight > 100){
                throw new InvalidProductException("Invalid Product:
                Weight cannot be greater than 100 grams");
            } else {
                p.display();
            }
        } catch (InvalidProductExcepion e){
            System.out.println(e.getMessage());
        }
    }
}
```

2.

```
import java.util.Scanner;

class InvalidEmailException extends Exception{
    InvalidEmailException(String message) {
        super(message);
    }
}
```

```
Public class Q2{
  public static void main(String[] args){
    Scanner sc= new Scanner(System.in);
    String email;

    System.out.print("Enter Email Address:");
    email = sc.nextLine();

    try{
      if (email.matches("^[0-9].*")){
        throw new InvalidEmailException("Invalid Email:
        Email address cannot start with a digit");
      }else if(!email.contains("@")){
        throw new InvalidEmailException("Invalid Email:
        Email Address must contain @ symbol.");
      }else{
        System.out.println("Valid email address:" + email);
      }
    }catch(InvalidEmailException e){
      System.out.println(e.getMessage());
    }
  }
}
```

3.

```
import java.util.Scanner;

class Item{
    String itemcode, description;
    int quantity;
    double rate;
```

```
public Item(String itemCode, String description, int quantity,
            double, rate) throws Exception {
    if (quantity <= 0) {
        throw new Exception("Quantity cannot be less than or
                             equal to zero");
    }
    if(rate <=0){
        throw new Exception("Rate cannot be less than or equal
                             to zero");
    }
    this.itemcode =itemCode;
    this.description= description;
    this.quantity = quantity;
    this.rate = rate;

    @Override
    public String toString(){
        return "Item [itemCode =" + itemCode + ", description =" +
        description + ", quantity=" + quantity + ", rate =" + rate
        + "]";
    }
}

public class Q3 {
    public static void main(String [] args) {
        Scanner sc = new Scanner(System.in);
        Item [] items = new Item [4];
        for (int i=0; i< items.length; i++) {
            try {
                System.out.println("Enter details of item " + (i+1));
                System.out.print("Item Code: ");
                String itemCode = sc.next();
```

```
            System.out.print("Description: ");
            String description= sc.next();
            System.out.print("Quantity:");
            int quantity = sc.nextInt();
            System.out.print("Rate: ");
            double rate = sc.nextDouble();
            items[i] = new Item(itemCode, description, quantity, rate);
        }catch (Exception e){
            System.out.println("Error: "+ e.getMessage());
            i--;
        }
    }
    System.out.println("Details of accepted items: ");
    for( Item item: items){
        System.out.println(item);
    }
  }
}
```

4.

```
import java.util.Scanner;

public class Q4 {
    public static void main(String[] args){
        Scanner sc= new Scanner(System.in);
        System.out.print("Enter username:");
        String username = sc.nextLine();
        System.out.print("Enter password :");
        String password =sc.next();
        try{
            if(!username.equals(password)){
                throw new Exception("Invalid Password");
```

```
        }
        System.out.println("Login successful!");
    } catch (Exception e) {
        System.out.println("Error: " + e.getMessage());
    }
}
}
```

5.

```
public class Q5
    Public static void main(String[] args) {
        if(args.length == 0) {
            System.out.println("Please enter a number as command
                                line argument");
            return;
        }
        try {
            if (number % 7 != 0){
                throw new Exception("not Divisible By 7");
            }
            System.out.println(number + "is divisible by 7");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}
```

6.

```
Public class Q6{
    public static void main(String[] args) {
        if (args.length == 0) {
            System.out.println("Please enter a string as
                                a command line argument");
```

```
            return;
        }
        String str = args[0];
        try {
            if (str.equals(str.toUpperCase())) {
                throw new UppercaseException();
            }
            System.out.println(str + " is not in uppercase");
        } catch (UpperCaseException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

class UpperCaseException extends Exception {
    public UpperCaseException() {
        super("Input string is in uppercase");
    }
}
```

7.

```
import java.util.Scanner;

public class Q7 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a positive integer: ");
        int num = sc.nextInt();
        try {
            if (num < 0) {
                throw new NegativeValueException();
            }
```

```
            System.out.println(num + " is a positive integer");
        } catch(NegativeValueException e) {
            System.out.println("Error: " + e.getMessage());
        }
    }
}

class NegativeValueException extends Exception {
    public NegativeValueException() {
        super("Negative value entered");
    }
}
```

8.

	Error	Exception
1	The error indicates trouble that primarily occurs due to scarcity of system resources	The exception are the issues that can appear at runtime and compile time.
2	It is not possible to recover from an error All the errors are unchecked	Its possible to recover from an exception The exceptions can be both checked and unchecked.

9.

Yes, it is possible to have an empty catch block in Java. However as I mentioned earlier, it is generally not considered good coding practice for the reasons In Java, the catch block is used to handle exception that may be thrown by the try block. The reason it is discouraged is that an empty catch block essentially means that the program is ignoring an exception

that has occured, and it provides no information or indication to the developer or user of the program about what went wrong or how to fix it.

10. ~~What is the d~~

Throw is used to throw an exception explicitly in the code. When an exception is thrown using the 'throw' keyword the program execution is stopped and the execution is propogated up the call stack until it is caught by an appropriate catch block that handles the exception. Throws is used in method declaration to indicate that ~~handles~~ the method may throw a particular type of exception. When a method throws an exception using 'throws' it is indicating to the calling code that the method may potentially fail and throw an exception of that type.