by Kunal Sir

Assignment: Exception Handling (try and catch block)

Part A: MCQs (20 Questions)

1. What is an exception in Java?

A. Syntax mistake
B. Compile-time warning
C. Abnormal condition that disrupts program flow
D. JVM shutdown

Answer: C

2. Which keyword is used to handle exceptions?

A. handle
B. error
C. catch
D. fix

Answer: C

3. Which block must be written before `catch`?

A. finally
B. throw
C. try
D. throws

Answer: C

4. Which of the following is NOT a reason for exception?

A. Invalid user input
B. Device failure
C. Network issue
D. Correct logic

Answer: D

---

5. Which package contains Exception class?

A. java.util
B. java.io
C. java.lang
D. java.net

Answer: C

---

6. Which exception occurs when dividing by zero?

A. NullPointerException
B. ArithmeticException
C. IOException
D. NumberFormatException

Answer: B

---

7. Which block always executes whether exception occurs or not?

A. try
B. catch
C. throw
D. finally

Answer: D

---

8. How many catch blocks can follow a try block?

A. Only one
B. Only two
C. Multiple
D. None

Answer: C

---

9. Which exception occurs when accessing array out of range?

A. ArrayIndexOutOfBoundsException
B. NullPointerException
C. ClassNotFoundException
D. IOException

Answer: A

---

10. Which keyword is used to manually create an exception?

A. throws
B. throw
C. catch
D. finally

Answer: B

---

11. Which keyword is used in method declaration?

A. throw
B. try
C. throws
D. catch

Answer: C

---

12. Checked exceptions are checked at:

A. Runtime
B. Compile-time
C. Execution end
D. JVM shutdown

Answer: B

---

13. Which is an unchecked exception?

A. IOException
B. SQLException
C. ClassNotFoundException
D. NullPointerException

Answer: D

---

14. Errors are mainly caused by:

A. User input
B. Coding mistakes
C. System resources
D. Logic errors

Answer: C

---

15. Can we handle Errors using try-catch?

A. Yes
B. No
C. Sometimes
D. Only runtime errors

Answer: B

---

16. Which is true about Error?

A. Recoverable
B. Must be handled
C. Outside program control
D. User-defined

Answer: C

---

17. Which exception occurs when file is not found?

A. IOException
B. FileNotFoundException
C. ArithmeticException
D. RuntimeException

Answer: B

---

18. What happens if exception is not handled?

A. Program continues
B. Program terminates
C. JVM ignores it
D. Only warning shown

Answer: B

---

19. Which block contains risky code?

A. catch
B. finally
C. try
D. throws

Answer: C

---

20. What is the parent class of all exceptions?

A. Error
B. Throwable
C. Object
D. Exception

Answer: B

Part B: Problem Statements

1. ATM Withdrawal System

Problem Statement:
Write a Java program for an ATM machine. The program should ask the user to enter the account balance and the withdrawal amount.
If the withdrawal amount is greater than the balance or the user enters an invalid number, the program should handle the problem using exception handling and show a proper message instead of crashing.

Sample Input:

```
Enter account balance: 5000
Enter withdrawal amount: 6000
```

Sample Output:

```
Error: Insufficient balance.
Transaction failed safely.
```

```java
import java.util.*;
import java.io.*;
class ATMWithdrawal {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter account balance: ");
            int balance = sc.nextInt();

            System.out.print("Enter withdrawal amount: ");
            int withdraw = sc.nextInt();

            if (withdraw > balance) {
                throw new Exception("Insufficient balance.");
            }

            System.out.println("Withdrawal successful.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.out.println("Transaction failed safely.");
        }
    }
}
```

2. Student Percentage Calculator

Problem Statement:
Create a Java program that takes total marks and obtained marks from the user and calculates the percentage.
If the user enters zero or invalid input, the program should handle the exception and show a clear message.

Sample Input:

```
Enter total marks: 0
Enter obtained marks: 450
```

Sample Output:

```
Error: Total marks cannot be zero.
Please enter valid marks.
```

```
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter total marks: ");
            int total = sc.nextInt();

            System.out.print("Enter obtained marks: ");
            int obtained = sc.nextInt();

            if (total == 0) {
                throw new ArithmeticException("Total marks cannot be zero.");
            }

            double percentage = (obtained * 100.0) / total;
            System.out.println("Percentage: " + percentage);
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.out.println("Please enter valid marks.");
        }
    }
}
```

3. User Login System

Problem Statement:
Write a Java program for a simple login system. The user enters a username and password.
If the username or password is empty or null, handle the exception and show a friendly message.

Sample Input:

```
Username:
Password: admin123
```

Sample Output:

```
Error: Username cannot be empty.
Login failed.
```

```java
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter username: ");
            String username = sc.nextLine();

            System.out.print("Enter password: ");
            String password = sc.nextLine();

            if (username == null || username.isEmpty()) {
                throw new Exception("Username cannot be empty.");
            }

            if (password == null || password.isEmpty()) {
                throw new Exception("Password cannot be empty.");
            }

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

4. File Reading Program

Problem Statement:
Create a Java program that reads data from a file name entered by the user.
If the file does not exist, the program should handle the exception and display a message instead of stopping.

Sample Input:

```
Enter file name: data.txt
```

Sample Output:

```
Error: File not found.
Please check the file name.
```

```
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter file name: ");
            String fileName = sc.nextLine();

            FileReader fr = new FileReader(fileName);
            BufferedReader br = new BufferedReader(fr);

            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
            }
            br.close();
        } catch (FileNotFoundException e) {
            System.out.println("Error: File not found.");
            System.out.println("Please check the file name.");
        } catch (IOException e) {
            System.out.println("Error reading file.");
        }
    }
}
```

5. Online Payment System

Problem Statement:
Write a Java program that accepts a payment amount from the user.
If the user enters a negative amount or invalid input, the program should handle the exception and show an error message.

Sample Input:

```
Enter payment amount: -500
```

Sample Output:

```
Error: Payment amount must be positive.
Transaction cancelled.
```

```java
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter payment amount: ");
            int amount = sc.nextInt();

            if (amount <= 0) {
                throw new Exception("Payment amount must be positive.");
            }

            System.out.println("Payment successful.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.out.println("Transaction cancelled.");
        }
    }
}
```

6. Student Record Using Array

Problem Statement:
Create a Java program that stores student names in an array.
Ask the user to enter an index number to view a student name.
If the index is invalid, handle the exception and show a message.

Sample Input:

```
Enter index number: 5
```

Sample Output:

```
Error: Invalid index.
Student record not found.
```

```java
import java.util.*;
import java.io.*;

class StudentRecord {
    public static void main(String[] args) {
        String[] students = {"Amit", "Rohit", "Sneha", "Pooja"};

        Scanner sc = new Scanner(System.in);

        try {
            System.out.print("Enter index number: ");
            int index = sc.nextInt();

            System.out.println("Student Name: " + students[index]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Error: Invalid index.");
            System.out.println("Student record not found.");
        }
    }
}
```

7. Bank Account Creation

Problem Statement:
Write a Java program to create a bank account.
The user enters age and initial deposit amount.
If age is less than 18 or deposit is below minimum amount, handle the exception.

Sample Input:

```
Enter age: 16
Enter deposit amount: 1000
```

Sample Output:

```
Error: Age must be 18 or above.
Account creation failed.
```

```
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter age: ");
            int age = sc.nextInt();

            System.out.print("Enter deposit amount: ");
            int deposit = sc.nextInt();

            if (age < 18) {
                throw new Exception("Age must be 18 or above.");
            }

            if (deposit < 2000) {
                throw new Exception("Minimum deposit is 2000.");
            }

            System.out.println("Account created successfully.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.out.println("Account creation failed.");
        }
    }
}
```

8. Calculator Application

Problem Statement:
Create a Java calculator that performs division of two numbers.
If the user tries to divide a number by zero, handle the exception and show a proper message.

Sample Input:

```
Enter first number: 10
Enter second number: 0
```

Sample Output:

```
Error: Cannot divide by zero.
Please enter a valid number.
```

```
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter first number: ");
            int a = sc.nextInt();

            System.out.print("Enter second number: ");
            int b = sc.nextInt();

            int result = a / b;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Cannot divide by zero.");
            System.out.println("Please enter a valid number.");
        }
    }
}
```

9. Railway Ticket Booking System

Problem Statement:
Write a Java program for booking railway tickets.
Ask the user how many seats they want to book.
If seats are not available or input is invalid, handle the exception.

Sample Input:

```
Enter number of seats: -2
```

Sample Output:

```
Error: Invalid seat number.
Booking failed.
```

```java
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter number of seats: ");
            int seats = sc.nextInt();

            if (seats <= 0) {
                throw new Exception("Invalid seat number.");
            }

            System.out.println("Tickets booked successfully.");
        } catch (Exception e) {
            System.out.println("Error: " + e.getMessage());
            System.out.println("Booking failed.");
        }
    }
}
```

10. Mobile Recharge System

Problem Statement:
Create a Java program for mobile recharge.
The user enters a recharge amount.
If the amount is less than the minimum recharge value or invalid, handle the exception properly.

Sample Input:

```
Enter recharge amount: 5
```

Sample Output:

```
Error: Minimum recharge amount is 10.
Recharge failed.
```

```java
import java.util.*;
import java.io.*;

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

        try {
            System.out.print("Enter recharge amount: ");
            int amount = sc.nextInt();

            if (amount < 10) {
                throw new Exception("Minimum recharge amount is 10.");
            }

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