JAVA PROGRAMMING ASSIGNMENT

Name: Gisubizo Gisele

ID: 26188

Class: Java Sunday Class

Submission Date: March 30, 2025

1. Checked exceptions are checked at compile-time and must be caught or declared with 'throws'. Example: IOException (e.g., FileReader file = new FileReader("file.txt");). Unchecked exceptions are not checked at compile-time and inherit from RuntimeException, usually indicating programming errors. Example: ArithmeticException (e.g., int x = 10 / 0;).

2. Output: Exception caught: / by zero

Explanation: The divide method attempts 10 / 0, which throws an ArithmeticException. The try-catch block catches it, and e.getMessage() prints "/ by zero".

3. The finally block runs code (e.g., closing resources) after try or catch, regardless of an exception. It executes after try completes or after catch handles an exception. It can be skipped if System.exit(0) is called or the JVM crashes.

4. 'throws' declares that a method can throw exceptions, part of the method signature. 'throw' explicitly throws an exception in code. Example: public void myMethod() throws IOException { if (true) throw new IOException("Error"); }

5. Custom exception:

public class InsufficientFundsException extends Exception {

public InsufficientFundsException(String message) {

super(message);

}

}

BankAccount class:

public class BankAccount {

private double balance;

public BankAccount(double initialBalance) {

this.balance = initialBalance;

}

public void withdraw(double amount) throws InsufficientFundsException {

if (amount > balance) {

throw new InsufficientFundsException("Insufficient funds: " + amount + " exceeds balance " + balance);

}

balance -= amount;

System.out.println("Withdrawal successful. New balance: " + balance);

}

public static void main(String[] args) {

BankAccount account = new BankAccount(100);

try {

account.withdraw(150);

} catch (InsufficientFundsException e) {

System.out.println(e.getMessage());

}

}

}

6. Output:

Inside finally block

Exception in thread "main" java.lang.ArithmeticException: / by zero

Explanation: 10/0 throws ArithmeticException. The finally block runs before the exception terminates the program.

7. public class StringToInt {

public static int convertToInt(String str) {

try {

return Integer.parseInt(str);

} catch (NumberFormatException e) {

return -1;

}

}

public static void main(String[] args) {

System.out.println(convertToInt("123")); // 123

System.out.println(convertToInt("abc")); // -1

}

}

8. Custom exception:

public class InvalidAgeException extends Exception {

public InvalidAgeException(String message) {

super(message);

}

}

Student class:

public class Student {

private int age;

public void setAge(int age) throws InvalidAgeException {

if (age < 0) {

throw new InvalidAgeException("Age cannot be negative: " + age);

}

this.age = age;

System.out.println("Age set to: " + age);

}

public static void main(String[] args) {

Student student = new Student();

try {

student.setAge(-5);

} catch (InvalidAgeException e) {

System.out.println(e.getMessage());

}

}

}

9. import java.util.Scanner;

public class DivisionProgram {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int num1, num2;

System.out.print("Enter first number: ");

num1 = scanner.nextInt();

while (true) {

try {

System.out.print("Enter second number: ");

num2 = scanner.nextInt();

int result = num1 / num2;

System.out.println("Result: " + result);

break;

} catch (ArithmeticException e) {

System.out.println("Error: Division by zero. Enter a valid denominator.");

}

}

scanner.close();

}

}

10. import java.util.Scanner;

public class ArrayIndexProgram {

public static void main(String[] args) {

int[] array = {10, 20, 30, 40, 50};

Scanner scanner = new Scanner(System.in);

System.out.print("Enter an index (0-4): ");

try {

int index = scanner.nextInt();

System.out.println("Element: " + array[index]);

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("Error: Invalid index.");

}

scanner.close();

}

}

11. public class EmailValidator {

public static void validateEmail(String email) {

if (!email.contains("@") || !email.contains(".")) {

throw new IllegalArgumentException("Invalid email: Must contain '@' and '.'");

}

System.out.println("Valid email: " + email);

}

public static void main(String[] args) {

try {

validateEmail("test@example.com");

validateEmail("invalid-email");

} catch (IllegalArgumentException e) {

System.out.println(e.getMessage());

}

}

}

12. import java.util.Scanner;

import java.util.InputMismatchException;

public class IntegerInputProgram {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int number;

while (true) {

try {

System.out.print("Enter an integer: ");

number = scanner.nextInt();

System.out.println("You entered: " + number);

break;

} catch (InputMismatchException e) {

System.out.println("Error: Please enter a valid integer.");

scanner.next();

}

}

scanner.close();

}

}

13. public class MultipleExceptions {

public static void main(String[] args) {

String str = null;

int[] array = {1, 2, 3};

try {

System.out.println(str.length());

System.out.println(10 / 0);

System.out.println(array[5]);

} catch (NullPointerException e) {

System.out.println("NullPointerException: " + e.getMessage());

} catch (ArithmeticException e) {

System.out.println("ArithmeticException: " + e.getMessage());

} catch (ArrayIndexOutOfBoundsException e) {

System.out.println("ArrayIndexOutOfBoundsException: " + e.getMessage());

}

}

}