**Velammal College of Engineering and Technology, Madurai**

**An Autonomous Institution**

**Department of Computer Science and Engineering**

**21CS205 Object oriented programming lab**

**Exercise No 6**

Exception Handling

Team 1

An application named DistanceFromAverage that allows a user to enter up to 20 double values and then displays each entered value and its distance from the average. Now, modify that program to first prompt the user to enter a number that represents the array size. Java generates a NegativeArraySizeException if you attempt to create an array with a negative size, and it creates a NumberFormatException if you attempt to create an array using a nonnumeric value for the size. Handle these exceptions using a catch block that displays an appropriate message. If the array is created successfully, use exception-handling techniques to ensure that each entered array value is a double before the program calculates each element’s distance from the average. Save the file as DistanceFromAverageWithExceptionHandling.java.

Team 2

Create an ApartmentException class whose constructor receives a String that holds a street address, an apartment number, a number of bedrooms, and a rent value for an apartment. Save the file as ApartmentException.java. Create an Apartment class with those fields. The Apartment constructor requires values for each field. Upon construction, throw an ApartmentException if the apartment number does not consist of three digits, if the number of bedrooms is less than 1 or more than 4, or if the rent is less than $500 or over $2,500. Save the class as Apartment.java. Write an application that establishes an array of at least six Apartment objects with valid and invalid values. Display an appropriate message when an Apartment object is created successfully and when one is not. Save the file as ThrowApartmentException.java.

Team 3

Create a UsedCarException class that extends Exception; its constructor receives a value for a vehicle identification number (VIN) that is passed to the parent constructor so it can be used in a getMessage() call. Save the class as UsedCarException.java. Create a UsedCar class with fields for VIN, make, year, mileage, and price. The UsedCar constructor throws a UsedCarException when the VIN is not four digits; when the make is not Ford, Honda, Toyota, Chrysler, or Other; when the year is not between 1990 and 2014 inclusive; or either the mileage or price is negative. Save the class as UsedCar.java. Write an application that establishes an array of at least seven UsedCar objects and handles any Exceptions. Display a list of only the UsedCar objects that were constructed successfully. Save the file as ThrowUsedCarExceptions.java.

Team 4

Write an application that displays a series of at least five student ID numbers (that you have stored in an array) and asks the user to enter a numeric test score for the student. Create a ScoreException class, and throw a ScoreException for the class if the user does not enter a valid score (less than or equal to 100). Catch the ScoreException, and then display an appropriate message. In addition, store a 0 for the student’s score. At the end of the application, display all the student IDs and scores. Save the files as ScoreException.java and TestScore.java.

Team 5

Write an application that displays a series of at least 10 student ID numbers (that you have stored in an array) and asks the user to enter a test letter grade for the student. Create an Exception class named GradeException that contains a static public array of valid grade letters (‘A’, ‘B’, ‘C’, ‘D’, ‘F’, and ‘I’) that you can use to determine whether a grade entered from the application is valid. In your application, throw a GradeException if the user does not enter a valid letter grade. Catch the GradeException, and then display an appropriate message. In addition, store an ‘I’ (for Incomplete) for any student for whom an exception is caught. At the end of the application, display all the student IDs and grades. Save the files as GradeException.java and TestGrade.java

Team 6

Create a DataEntryException class whose getMessage() method returns information about invalid integer data. Write a program named GetIDAndAge that continually prompts the user for an ID number and an age until a terminal 0 is entered for both. Throw a DataEntryException if the ID is not in the range of valid ID numbers (0 through 999), or if the age is not in the range of valid ages (0 through 119). Catch any DataEntryException or InputMismatchException that is thrown, and display an appropriate message. Save the files as DataEntryException.java and GetIDAndAge.java.

Team 7

Create an application that accepts employee data interactively. Users might make any of the following errors as they enter data: The employee number is not numeric, less than 1000, or more than 9999. The hourly pay rate is not numeric, less than $9.00, or more than $25.00. Create a class that stores an array of six usable error messages that describe the preceding mistakes; save the file as EmployeeMessages.java. Create an EmployeeException class; each object of this class will store one of the messages. Save the file as EmployeeException.java. Create an application that prompts the user for employee data, and display the appropriate message when an error occurs. If no error occurs, display the message “Valid employee data”. Save the program as EmployeeDataEntry.java

Team 8

Create a CalculatorDemo program that asked the user to solve an arithmetic problem and provided the system calculator for assistance. Now modify that program to include the following improvements: Both numbers in the arithmetic problem should be random integers between 1 and 5,000. The program should ask the user to solve five problems. The program should handle any noninteger data entry by displaying an appropriate message and continuing with the next problem. Save the file as CalculatorDemo2.java