Assignment 07

1. Develop a piece of Java code that demonstrates the handlers for exceptions thrown by the code. The exceptions the code may throw along with the handler message are listed below:
    a. Division by zero: Print "Invalid division".
    b. String parsed to a numeric variable : Print "Format mismatch".
    c. Accessing an invalid index in string : Print "Index is invalid".
    d. Accessing an invalid index in array : Print "Array index is invalid".

   MyException : This is a user defined Exception which you need to create. It takes a parameter When an exception of this class is encountered, the handler should print "MyException[param]", here is the parameter passed to the exception class.

   Finally, after the exception is handled, print "Exception Handling Completed".

2. Write a Java program that creates a class hierarchy for employees of a company. The base class should be Employee, with subclasses Manager, Developer, and Programmer. Each subclass should have properties such as name, address, salary, and job title. Implement methods for calculating bonuses (salary*0.15), generating performance reports (excellent, avg, poor), and managing projects. (You have to involve different access specifiers and getter and setter methods wherever applicable, and assume your own employee details wherever required)

   Explanation:

   Employee class: This class represents a generic employee with private instance variables 'name', 'address', 'salary', and 'jobTitle'. It also provides getter methods to access these private variables.
   getName(): Returns the employee's name.
   getAddress(): Returns the employee's address.
   getSalary(): Returns the employee's salary.
   getJobTitle(): Returns the employee's job title.

   calculateBonus(): This method is used to calculate the bonus for an employee. In the base class, it provides a default implementation that returns 0.0. Subclasses can override this method to provide custom bonus calculation logic.

   generatePerformanceReport(): This method generates a performance report for an employee. Similar to the bonus calculation, it provides a default implementation that returns "No performance report available." Subclasses can override this method to provide custom performance report generation logic.

   Employee class is designed to be extended by subclasses like "Manager", "Developer", and "Programmer", which can provide their own implementations of bonus calculation and performance report generation as per their specific roles and responsibilities.
   Main class explanation:
   Creating Employee Objects:

Three employee objects are created: 'manager', 'developer', and 'programmer', each with their specific attributes such as name, address, salary, and job title.
manager is an instance of the "Manager" class.
developer is an instance of the "Developer" class.
programmer is an instance of the "Programmer" class.

Calculating Bonuses:

The program calls the "calculateBonus()" method for each employee type (manager, developer, and programmer) to calculate their respective bonuses. The bonuses are displayed on the console.

Generating Performance Reports:

The program calls the "generatePerformanceReport()" method for each employee type (manager, developer, and programmer) to generate performance reports. The performance reports are displayed on the console.

3. Write a Java program to create a vehicle class hierarchy. The base class should be Vehicle, with subclasses Truck, Car and Motorcycle. Each subclass should have properties such as make, model, year, and fuel type. Implement methods for calculating fuel efficiency, distance traveled, and maximum speed. Go through the explanation below and assume your own vehicle details wherever required.

Explanation:

The vehicle class is an abstract class that serves as the parent class for the other vehicle sub classes. It contains five private instance variables (make, model, year, fuelType, and fuelEfficiency) and six public methods (a constructor, five getters for the instance variables, and three abstract methods). The abstract methods are meant to be overridden by child classes with specific implementations.

The Truck class is a child class of Vehicle and extends the Vehicle class. It has an additional instance variable, cargoCapacity. The class has a constructor that accepts all the necessary parameters including cargo capacity. The class overrides the three abstract methods of the parent class and provides specific implementations of the methods.

The car class is another child class of Vehicle and extends the Vehicle class. It has an additional instance variable, numSeats. The class has a constructor that accepts all the necessary parameters including the number of seats. The class overrides the three abstract methods of the parent class and provides specific implementations of the methods.

The motorcycle is a child class of Vehicle and extends the Vehicle class. It has an additional instance variable, engineDisplacement. The class has a constructor that accepts all the necessary parameters. The class overrides the three abstract methods of the parent class and provides specific implementations of the methods.

The main class contains the main method. It creates instances of each vehicle type, sets their values, and then prints their respective details and calculations such as fuel efficiency, distance traveled, and max speed.