**Please ignore my previous submissions as they were submitted by mistake and hastily.**

**Assignment 1**

About Java:

Java is an object-oriented programming language developed by Sun Microsystems in the year 1995

Features of Java:

* Simple and Familiar
* Compiled and Interpreted
* Platform Independent
* Portable
* Architectural Neutral
* Object-Oriented
* Robust
* Secure
* Distributed
* Multi-threaded and Interactive
* High Performance
* Dynamic and Extensible

Java Program Lifecycle:

Java programs normally undergo four phases

– Edit

• Programmer writes program (and stores program on disk)

– Compile

• Compiler creates byte codes from program (.class)

– Load

• Class loader stores byte codes in memory

– Execute

• Interpreter: translates byte codes into machine language

Four Pillars of Object-Oriented Programming Language:

1. Encapsulation
2. Abstraction
3. Inheritance
4. Polymorphism

Encapsulation: Encapsulation is the process of binding methods with data into a single entity called class. This is done to protect the implementation details of a class from the outside world.

public class EncapTest {

private String name;

private String idNum;

private int age;

public int getAge() {

return age;

}

public String getName() {

return name;

}

public String getIdNum() {

return idNum;

}

public void setAge( int newAge) {

age = newAge;

}

public void setName(String newName) {

name = newName;

}

public void setIdNum( String newId) {

idNum = newId;

}

}

Abstraction: Abstraction is the process of hiding irrelevant or background details and showing only relevant information to the end users. For example: we only need to know how to run a car and not how the wires are connected inside a car.

// Abstract class

abstract class Animal {

// Abstract method (does not have a body)

public abstract void animalSound();

// Regular method

public void sleep() {

System.out.println("Zzz");

}

}

// Subclass (inherit from Animal)

class Pig extends Animal {

public void animalSound() {

// The body of animalSound() is provided here

System.out.println("The pig says: wee wee");

}

}

class Main {

public static void main(String[] args) {

Pig myPig = new Pig(); // Create a Pig object

myPig.animalSound();

myPig.sleep();

}

}

Inheritance: Inheritance is the process by which behaviour and properties of a super class can be inherited by deriving a base class from the existing class. The new class will contain combined features of both the classes. Inheritance introduces the concept of reusability that is we can reuse the properties of the existing class by simply deriving a new class from the base class. Therefore, inheritance is the process by which objects of one class acquire the properties of objects of another class. For example: We derive flying and non-flying bird from the class bird (hierarchical) which share common characteristics with the parent class bird and from non-flying and flying bird we derive penguin and robin (multilevel) which share common characteristics with the classes from which they are derived.

class Calculation {

int z;

public void addition(int x, int y) {

z = x + y;

System.out.println("The sum of the given numbers:"+z);

}

public void Subtraction(int x, int y) {

z = x - y;

System.out.println("The difference between the given numbers:"+z);

}

}

public class My\_Calculation extends Calculation {

public void multiplication(int x, int y) {

z = x \* y;

System.out.println("The product of the given numbers:"+z);

}

public static void main(String args[]) {

int a = 20, b = 10;

My\_Calculation demo = new My\_Calculation();

demo.addition(a, b);

demo.Subtraction(a, b);

demo.multiplication(a, b);

}

}

Types of Inheritance:

* + - 1. Single Inheritance: When a derived class inherits its properties and behavior from a single base class
      2. Multilevel Inheritance: When a derived class is created from another derived class
      3. Hierarchical Inheritance: When two or more derived classes are created from a single base class
      4. Hybrid Inheritance: It’s a combination of one or more inheritance
      5. Multiple Inheritance: When a derived class is created from one or more base classes

Polymorphism: It is the process by which the same method performs different tasks based on the object it is acting upon. For example: The method draw can be used to draw a box or circle or triangle.

public class Employee {

private String name;

private String address;

private int number;

public Employee(String name, String address, int number) {

System.out.println("Constructing an Employee");

this.name = name;

this.address = address;

this.number = number;

}

public void mailCheck() {

System.out.println("Mailing a check to " + this.name + " " + this.address);

}

public String toString() {

return name + " " + address + " " + number;

}

public String getName() {

return name;

}

public String getAddress() {

return address;

}

public void setAddress(String newAddress) {

address = newAddress;

}

public int getNumber() {

return number;

}

}

Overloading: When two or more methods with the same name but different parameters exist in a class it is known as method overloading. Method overloading is used when we want objects to perform similar tasks using different parameters. That is draw(previous example)

Overriding: When methods with the same name but different parameters exist in different classes such as parent and child classes then when the method is called, the methods in the child class gets invoked and executed first. That is the methods in the child class overrides the methods in the parent class. Method overriding is used when we want an object to respond to the same method but show different behavior when the method is called. Example: Draw a circle or triangle or box when the method draw is called.

Method Overloading is known as compile time polymorphism because it is resolved at compile time whereas method overriding is known as run time polymorphism because it is resolved at run time.

The Java programming language defines eight primitive

types:

• Logical – boolean

• Textual – char

• Integral – byte, short, int, and long

• Floating – double and float

Reference Data Types:

Reference variables are created using defined constructors of

the classes.

• Class objects, and various type of array variables come under

reference data type.

• Default value of any reference variable is null.

• A reference variable can be used to refer to any object of the

declared type or any compatible type.

– Example : Animal animal = new Animal("giraffe");

SDLC:

SDLC is a framework that describes the complete cycle of development that is all the tasks involved in planning, creating, testing and deploying a software product.

The stages in SDLC are:

* + 1. Requirement Analysis
    2. Specification
    3. Software Architecture
    4. Implementation
    5. Testing
    6. Documentation
    7. Training and Support
    8. Maintenance

SDLC models:

* 1. The Waterfall Model
  2. The Spiral Model
  3. V-model
  4. Agile

OOAD:

• A popular technical approach for analysing & designing an application, system, or business

• Used to create visual modelling throughout the development life cycles to foster better stakeholder communication and product quality

The purpose of OO analysis and design :

• Identifying the objects of a system.

• Identify their relationships.

• Make a design which can be converted to executables using OO languages.

Class:

Class is a blueprint of an object

Object:

Object is the instance of a class.It has its own behavior, state and identity.

The primary tasks in object-oriented analysis (OOA) are:

– Find the objects

– Organize the objects

– Describe how the objects interact

– Define the behavior of the objects

– Define the internals of the objects

Object Oriented Design:

o To solve a problem that was identified and documented during object-oriented analysis

o Object-oriented design is the discipline of o defining the objects &

their interactions

Role of UML in OO analysis & design:

o UML is a modelling language used to model software and non-software systems

o The OO analysis & design is transformed into UML diagrams according to the requirement

Properties of UML:

• UML stands for Unified Modelling Language.

• The language is very rich, and carries with it many aspects of Software Engineering best practice

• UML is different from the other common programming languages like C++, Java, COBOL etc.

• UML is a pictorial language used to make software blue prints.

• The UML is a graphical language for capturing the artifacts of software developments.

• The language provides us with the notations to produce models.

• The UML is adopted industry wide language.

• The UML was originally designed by the Three Amigos at Rational Corp.

Why UML?

• The Unified Modelling Language (UML) offers a way to visualize a system's architectural blueprints in a diagram

– Any activities (jobs)

– Individual components of the system

• And how they can interact with other software components.

– How the system will run

– How entities interact with others (components and interfaces)

– External user interface

Types of Diagrams:

• Use Cases - How will our system interact with the outside world?

• Class & Object Diagram - What objects do we need? How will they be related?

• Collaboration Diagram - How will the objects interact?

• Sequence Diagram - How will the objects interact(order/time)?

• State Diagram - What states should our objects be in?

• Package Diagram - How are we going to modularize our development?

• Component Diagram - How will our software components be related?

• Deployment Diagram - How will the software be deployed?

What is a conceptual model?

– It helps to understand the entities in the real world and how they interact with each other.

– Conceptual Modelling (sometimes called Domain Modelling) is the activity of finding out which concepts are important to our system

• On the conceptual model, we aim to capture all of the concepts or ideas that the customer recognizes.

• For example, some good examples of concepts would be:

– Lift in a lift control system

– Order in a home shopping system

– Footballer in a PlayStation football game

– Trainer in a online e-learning system

– Room in a room booking system

• Event Trigger - the special process that waits for 5 minutes and then tells the system to wake up and do something

• CustomerDetailsForm - the window that asks for details of the new customer in a shopping system

• DbArchiveTable - the database table holding a list of all old orders

UML Diagrams:

• Use case diagrams

• Class diagrams

• Object diagrams

• Sequence diagrams

• Collaboration diagrams

• State chart diagrams

• Activity diagrams

• Component diagrams

• Deployment diagrams

**Assignment 2**

What is a method?

Method is an action which an object is able to perform

Creating a method :

public int funcName(int a, int b) { // body }

public : Access specifier

int: return type

funcName: function name

int a, int b: list of parameters

What is a constructor?

A constructor is a special method that initializes an object and allocates memory to it. A constructor is automatically invoked when an object is created.

What are parameters and argument?

**Parameter** is variable defined in function definition, while **argument**is actual value passed to the function.

What are the different types of constructors?

Default constructor: Doesn’t have any parameter

Parameterized constructor: Takes in parameters

Copy Constructor: Copy Constructor is a member function that initializes and object using another object of the same class.

What are the different types of operators?

Arithmetic Operators: C Arithmetic operators are used to perform mathematical calculations like addition, subtraction, multiplication, division and modulus in C programs.

Arithmetic operators are +, -, \*, /, %

Assignment Operators: Assignment operators are used to assign the values to the variables.

Assignment operators are =, +=, -=, /+, %= etc.

Relational Operators: Relational operators are used to find the relation between two variables. i.e. to compare the values of two variables in a C program.

Relational operators are >, <, >=, <=, ==, !=

Logical Operator: Logical operators are used to perform logical operations on the given expressions.

There are 3 logical operators in C language. They are,

1. logical AND (&&)
2. logical OR (||)
3. logical NOT (!)

Bitwise Operator: Bitwise operators are used to perform bit operations. Decimal values are converted into binary values which are the sequence of bits and bit wise operators work on these bits.

Bit wise operators in C language are & (bitwise AND), | (bitwise OR), ~ (bitwise OR), ^ (XOR), << (left shift) and >> (right shift).

Ternary Operator:

Ternary operators also known as conditional operators are operators that take three arguments

expression one? expression two: expression three

What is a variable?

Variable is a storage location that is allocated some memory. Different types of variables require different amounts of memory and have some specific set of operations that can be applied on them.

What are local variables?

Local variables are variables that are declared inside a function/method block.Their scope/visibility is limited to the function/method block in which they are declared.Lifetime of a local variable exists until the execution of the function/method block in which they are declared.

public class Test{

public void Age(){

int age = 0;

age = age + 7;

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

}

public static void main(String args[]){

Test test = new Test();

test.Age();

}

}

What are static variables?

Static variables are variables that declared using the keyword static.Their scope/visibility is limited to the function/method block in which they are declared however their values persists between function calls.

String:

String is a sequence of characters enclosed in single or double quotes

String Concatenation:

Example: string1 + string2

What are access specifiers?

Access specifiers are special keywords which control or specify the accessibility of entities like classes, methods etc. Some of the access specifiers or modifiers include public, private, protected. They play an important role in achieving encapsulation.

Access Control Modifiers:

1. Visible to the package, the default. No modifiers are needed.
2. Visible to the class only (private).
3. Visible to the world (public).
4. Visible to the package and all subclasses(protected).

Non Access Modifiers:

1. The static modifier for creating class methods and variables
2. The final modifier for finalizing the implementations of classes, methods, and variables.
3. The abstract modifier for creating abstract classes and methods.
4. The synchronized and volatile modifiers, which are used for threads.

Instance Variable:

* Declared in a class, but outside a method, constructor or any block.
* When space is allocated for an object in the heap, a slot for each instance variable value is created.
* Created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
* Hold values that must be referenced by more than one method or essential parts of an object's state that must be present throughout the class.
* Access modifiers can be given for instance variables.
* Instance variables have default values.

1. for numbers the default value is 0
2. for Booleans it is false
3. for object references it is null.

* Can be accessed directly by calling the variable name inside the class

**Assignment 3**

What is Exception?

Exception is a special event that is raised during execution of a program at run time which brings the execution of the program to a halt. The main reason for exception is an undesirable input.

What is Exception Handling?

Exception Handling is an important concept to manage errors. An Exception Handler allows errors to be thrown and caught and implements a centralized mechanism to resolve them. Try-catch block is one of the most popular methods for handling exceptions.

The Try block defines a set of statements that may lead to an error and the catch block basically catches the error.

public class JavaExceptionExample{

public static void main(String args[]){

try{

//code that may raise exception

int data=100/0;

}catch(ArithmeticException e){System.out.println(e);}

//rest code of the program

System.out.println("rest of the code...");

}

}

Types Of Exception:

* Checked Exception
* Unchecked Exception

1. Error
2. Runtime Exception

.

Checked:

* Exceptional conditions that a well-written application should anticipate and recover from
* Example : File Reading process

1. An application prompts a user for an input file name,then opens the file by passing the name.
2. The user provides the name of an existing, readable file, and the execution of the application proceeds normally.
3. If the user supplies the name of a nonexistent file an exception occurs

* A well-written program will catch this exception and notify the user of the mistake

Error:

* Exceptional conditions that are external to the
* application, and that the application usually cannot
* anticipate or recover from
* Example : File processing

1. An application successfully opens a file for input, but is unable to read the file because of a hardware or system malfunction.
2. the unsuccessful read will throw Error.

* An application might choose to catch this exception, in order to notify the user of the problem
* But it makes sense for the program to print a stack trace and exit.

Runtime Exception:

* These are exceptional conditions that are internal to the application, and that the application usually cannot anticipate or recover from.
* These usually indicate programming bugs, such as logic errors or improper use of an packages.
* Example : File Processing

1. In the file reading application, if a logic error causes a null to be passed it will cause an Exception.

* The application can catch this exception, but it probably makes more sense to eliminate the bug that caused the exception to occur.

Finally Block:

A finally block will always be executed regardless of whether an exception occurs or not.

class TestFinallyBlock {

public static void main(String args[]){

try{

//below code do not throw any exception

int data=25/5;

System.out.println(data);

}

//catch won't be executed

catch(NullPointerException e){

System.out.println(e);

}

//executed regardless of exception occurred or not

finally {

System.out.println("finally block is always executed");

}

System.out.println("rest of phe code...");

}

}