Object Oriented Programming –Assignment 01 (Encapsulation)

01. Briefly explain of followings terms with appropriate examples.

a. Class/ Template - A claaa or a template is a blueprint or a set of instructions that defines the structure and behavior of objects.
(ex:- Car,Employee)

b. Object/ Instance - An object or instance is a concrete reprecentation of a class.It is a variable that is created based on the class blueprint and holdsreal data.

c. Methods/ Function - Methods are functions defined within a class that perform actions or operations on the object's data.

d. Attributes/ Properties - Attributes, also called properties, are variables associated with a class and are used to store data that represents the object's state.

e. Reference Variables - Reference variables are variables that hold memory addresses pointing to the location of an object in memory rather than the actual data.

f.Primitive Variables - Primitive variables are variables that hold simple, basic data types such as integers, floating-point numbers, characters, or booleans.

g. Method Parameters - Method parameters are variables defined in a method's signature (declaration) that are used to pass data to the method when it is called.

h. Local Variables - Local variables are variables declared within a specific block of code, typically inside a method or a function

i. Default Values - Default values are initial values assigned to variables, parameters, or attributes when they are declared.
(int=0,double=0.0,String=null)

j. Declaration Values - Declaration values refer to the values assigned to variables, parameters, or attributes when they are declared in the code.
(ex: int num=0;)

02)
Given Code:

```
//--------------------Test.java------------------------
class Test{
int a=8;
int b=9;
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
System.out.print(a);
System.out.println(b);
}
}
```

What is the result of attempting to compile & run the program?

D. Compile time error

03. Given Code:

```
//--------------------Test.java------------------------
class Test{
int a=8;
int b=9;
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
Test t1=new Test();
System.out.print(t1.a);
System.out.println(t1.b);
}
}
```

What is the result of attempting to compile & run the program?

A. 89

04. Briefly explain the difference between “attributes” and “methods” in java Object using real world examples.

Attributes:

Attributes are also known as member variables, properties, or instance variables.

They represent the data or state of an object and define what the object is or what it possesses.

Attributes store information about the object's characteristics.

```
 ex: class Employee{
        String empId;
        String name;
        double salary;
 }
```

Methods:

Methods, on the other hand, are functions associated with a class that define the behaviors or actions that objects of that class can perform.

```
ex: public void work(){
            System.out.println(“Employee can work”);
}
```

05. Which of the following lines are illegal? Explain your answer.

```
class Box{
int length;
int width;
int height;
}

class Demo{
```

```
public static void main(String args[]){
Box b1=new Box(); // 1
System.out.println("length of box : "+length); //2
System.out.println("width of box : "+width); //3
System.out.println("height of box : "+height); //4
          }
}
```

1-legal
2-illegal
3-illegal
4-illegal

06. What is the output of the following program? Explain your answer.

```
//--------------------Box.java------------------------
class Box{
int length;
int width;
int height;
void printVolume(){
int volume;
volume=length*width*height;
System.out.println("Volume : "+volume);
          }
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
b1.printVolume();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
          }
}
```

output,

```
Volume : 0
length of box : 0
width of box : 0
height of box : 0
```

Since values are not assigned.default values are printed,according to the corresponding data type.

07. What is the output of the following program? Explain your answer.

```
//--------------------Box.java------------------------
class Box{
int length=12;
int width=5;
int height=3;
void printVolume(){
int volume;
volume=length*width*height;
System.out.println("Volume : "+volume);
}
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
b1.printVolume();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
}
}
```

output,

Volume : 180
length of box : 12
width of box : 5
height of box : 3
As values are assigned,corresponding values are printed.The Box class has a method called printVolume(), which calculates the volume of the box (length × width × height) and prints it to the console.

08. What is the output of the following program? Explain your answer.

```
//--------------------Box.java------------------------
class Box{
int length;
int width;
int height;
Box(){
System.out.println("default constructor");
length=2;
width=2;
height=2;
}
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
        }
}
```

output,

default constructor
length of box : 2
width of box : 2
height of box : 2
The constructor Box() is called when the object b1 is created. This constructor initializes the length, width, and height variables to 2 and prints "default constructor" to the console.

09. What is the output of the following program? Explain your answer.

```
//--------------------Box.java------------------------
class Box{
int length;
int width;
int height;
Box(int l,int w,int h){
System.out.println("Parameterized constructor");
length=l;
width=w;
height=h;
}
void printVolume(){
int volume;
volume=length*width*height;
System.out.println("Volume : "+volume);
}
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
Box b1=new Box(5,4,3);
b1.printVolume();
Box b2=new Box(12,5,3);
b2.printVolume();
}
}
```

output,

```
Parameterized constructor
Volume : 60
Parameterized constructor
Volume : 180
```

10. What is the output of the following program? Explain your answer.

```
class Box{
int length;
private int width;
int height;
}
//--------------------Demo.java------------------------
class Demo{
public static void main(String args[]){
Box b1=new Box();
System.out.println("length of box : "+b1.length);
System.out.println("width of box : "+b1.width);
System.out.println("height of box : "+b1.height);
        }
}
```

error: width has private access in Box
System.out.println("width of box : "+b1.width);
not output

11. Which of the following lines are illegal? Explain your answer.

```
class Box{
int length; //Line 1
int width; //Line 2
int height; //Line 3
length=12; //Line 4
width=5; //Line 5
height=3; //Line 6
}
```

line 4,line 5,line 6 are illegal.
you cannot have direct assignments to instance variables outside of methods or constructors. Instead, you should initialize the variables either during their declaration or inside a constructor.

```
class Box{
int length; //Line 1
int width; //Line 2
int height; //Line 3
        Box(){
        length=12; //Line 4
        width=5; //Line 5
        height=3; //Line 6
        }
}
```

12. What is the purpose of the constructor in a java Template (Class)? Explain your answer the suitable examples

a constructor is a method of a class that is responsible for initializing objects of that class. It is automatically called when you create an instance (object) of the class using the new keyword.

13. Briefly explain the difference between “constructor s” and “methods” in java Object using real world examples.

Constructor-
In oop ,A Constructor is a special method that is automatically called when as object is created using the “new keyword”.

Every class has a Constructor,that constructor gets the full responsibility to look at the class and create the object.

ex:-

```
class Student {
String id;
double prfMark;
double dbmMark;
boolean status;
//constructor
Student() {
// dan meya fully response eka gannawa to create a new Student object
}
}

class Main {
public static void main(String args[]) {
Student s1 = new Student(); //constructor
}
}
```

Methods-
In OOP,Behaviours are methos.
Methods are functions within a class that define the behavior of the objects created from that class

ex:-

```
class Employee {
String empId;
String name;
double salary;
String email;

public void work() { //Behaviours
System.out.println("Employee is working");
}
public void eat() {
System.out.println("Employee is eating");
}
}
```

14. Which of the following lines of code could be inserted at line 12 and still allow the code to compile?

```
class Box{
int length;
int width;
int height;
}
class Demo{
public static void main(String args[]){
//Insert code here
}
}
```

B. Box b1=new Box();

15. What is the output of following program? Explain your

```
class Box{
int length;
int width;
}
class Demo{
public static void main(String args[]){
Box b1=new Box();
System.out.print(b1.length); //Line 1
System.out.print(b1.WIDTH); //Line 2
System.out.print(b1.height); //Line 3
        }
}
```

not compile this programme.line 2,line 3 are false.
line 3-can't find height variable.

16. What is the output of the following program?

```
class Test{
int a=3;
int b=4;
Test(int i,int j){a=i;b=j;}
Test(){}
}
class Demo{
public static void main(String args[]){
Test t1=new Test(1,2);
System.out.print(t1.a+" "+t1.b+" ");
Test t2=new Test();
System.out.println(t2.a+" "+t2.b);
}
}
```

C. 1 2 3 4

17. Which of the following lines of code could be inserted at line 10 to get the following output?

Prints "Values 100 200"

```
class MyClass{
/* * Insert Code Here Line 10 */
void printValues(){
System.out.println("Values : "+x+", "+y);
        }
}
class Demo{
public static void main (String []args){
MyClass c=new MyClass(100,200);
c.printValues();
        }
}
```

A. int x;
int y;
MyClass(int i, int j){ x=i; y=j; }

B. int x=100;
int y=200;
MyClass(int i, int j){ }

D. int x;
int y;
MyClass(int i, int j){ x=100; y=200; }

18. What will be the output when you compile and run the following program? Explain your answer.

```
class A{
int a;
A(int i){ a=i; }
}
class Demo{
public static void main(String args[]){
A a1=new A(100);
A a2=new A(101);
System.out.println(a1.a+" "+a2.a);
a1=a2;
a1.a=0;
System.out.println(a1.a+" "+a2.a);
}
}
```

output,

```
100 101
0 0
```

In the first print statement, we get "100 101" because “a1.a is 100” and “a2.a is 101” initially.

After setting a1.a = 0, both a1.a and a2.a become 0 because they are referring to the same object, and changing a1.a also changes a2.a.(a1=a2)

19. What will be the output when you compile and run the following program? Explain your answer.

```
class Item{
int code;
Item(int i){ code=i;}
void printCode(int code){
System.out.println("Code : "+code);
}
}
class Demo{
public static void main(String args[]){
Item t1=new Item(2001);
t1.printCode(3001);
}
}
```

output,

```
Code : 3001
```

The argument passed to the printCode method is “code”.If “i” is passed as argument,2001 will be printed.

20. What will be the output when you compile and run the following program? Explain your answer.

```
class Item{
int code;
Item(int i){ code=i;}
void printCode(int code){
System.out.println("Code : "+this.code);
}
}
class Demo{
public static void main(String args[]){
Item t1=new Item(2001);
t1.printCode(3001);
}
}
```

output,
Code : 2001

21)
B. void A(int i){a=i;}
C. A(){}

22)
A. A(int i){a=i;}
D.A(int a){this.a=a;}
E. A(int i){a=100;}

23)
The properties and behaviours required to perform a specific or several tasks are gathered in one place and wrapped as a single unit to provide validational protection.It is called **Encapsulation.**

ex:-
phone-A phone has specific tasks (talking calls,sending messages,google search,listening songs). There are properties and behaviours that are required to perform these tasks(display,chip,battery,mother board,storage).All these have been gathered in one place ,wrapped with a plastic cover and according to a validational protection ,a single unit called the phone has been created.

24)
- **Loosely Encapsulated:**
In a loosely encapsulated class, the internal data (attributes) of the class may not be adequately protected, and it might be directly accessible from outside the class.
ex:-

```
//A is a Loosley Encapsulated class
class A {
int a;
String description;
}
```

- **Tightly Enacasulated:**
The class's data members are usually declared as private, preventing direct external access.
ex:-

```
// A is a Tighlty Encapsulated class
class A {
private int a;
private String description;
}
```

25)
D. Compiler error at line 2

26)
```
class Date{
	int year=1970;
	int month=1;
	int day=1;

	public void setYear(int i){
		year=i;
	}

	public void setMonth(int n){
		month=n;
	}

	public void setDay(int d){
		day=d;
	}

	public int getYear(){
		return year;
	}

	public int getMonth(){
		return month;
	}

	public int getDay(){
		return day;
	}

	public void printDate(){
		System.out.println(year+"-"+month+"-"+day);
	}
}

class pavi{
public static void main(String args[]){

	Date d1=new Date();
	d1.printDate();
	d1.year=2016; //illegal
	d1.month=5; //illegal
	d1.day=30;  //illegal

	d1.setYear(2016);
	d1.setMonth(5);
	d1.setDay(31);

	System.out.println("Year : "+d1.getYear());
	System.out.println("Month :"+d1.getMonth());
	System.out.println("Day : "+d1.getDay());
	}
}
```

output,

```
1970-1-1
Year : 2016
Month :5
Day : 31
```

27)

Explain the following lines of code according to class "Customer" is given bellow.

```
class Customer{
private String id;
private String name;
public Customer(){}
public Customer(String id, String name){
this.id=id;
this.name=name;
}
public void printCustomer(){
System.out.println(id+" - "+name);
}
public void setCustomerDetail(String id, String
name){
this.id=id;
this.name=name;
}
public void setId(String id){
this.id=id;
}
public void setName(String name){
this.name=name;
        }
}
//---------------------DemoCustomer.java--------------------
class DemoCustomer{
public static void main(String args[]){
Customer c1; //Line 1
c1=new Customer("C001", "Danapala");//Line 1
c1.printCustomer(); //Line 3
Customer c2; //Line 4
c2=new Customer(); //Line 5
c2.setCustomerDetail("C002", "Gunapala"); //Lin6
c2.printCustomer(); //Line 7
Customer c3; //Line 8
c3=new Customer(); //Line 9
c3.setId("C003"); //Line 10
c3.setName("Somapala"); //Line 11
c3.printCustomer(); //Line 12
        }
}
```

- ❖ **Customer c1; //Line 1** - Create a variable called "c1".It's data type is "Customer".
- ❖ **c1=new Customer("C001", "Danapala");//Line 2 -** Initializing a new Customer Object with values.

- **c1.printCustomer(); //Line 3 -** create a method that "printCustomer" to print c1 Customer's values.
- **Customer c2; //Line 4 -** Create a variable called "c2".It's data type is "Customer".

- **c2=new Customer(); //Line 5 -** Initializing a new Customer Object

- **c2.setCustomerDetail("C002", "Gunapala"); //Line6 -** create the method setCustomerDetail();,assign the c2 Customer's values.

- **c2.printCustomer(); //Line 7 -** print the values of c2 Customer ,using the printCustomer method.

- **Customer c3; //Line 8 -** Create a variable called "c3".It's data type is "Customer".

- **c3=new Customer(); //Line 9 -**Initializing a new Customer Object

- **c3.setId("C003"); //Line 10 -** set c3 Customer's id is "C003".

- **c3.setName("Somapala"); //Line 11 -** set c3 Customer's name is "Somapala".
- **c3.printCustomer(); //Line 12 -** print the values of c3 Customer ,using the printCustomer method.

28)
A. Compile Error at line 1
B. Compile Error at line 2

29)
F. None of the above

output is,
1 200 10 200 100 200

30)
```
class Rectangle{
        double length=10.0;
        double width=5.0;

        Rectangle(){}

        public void setLength(double x){
                length=x;
        }
        public void setWidth(double y){
                width=y;
        }

        public double getLength(){
                return length;
        }
        public double getWidth(){
                return width;
        }
```

```
	public void printPerimeter(){
		double perimeter=0;
		perimeter=length+length+width+width;

		System.out.println("Rectangle Perimeter is: "+perimeter);
	}

	public void printArea(){
		double area=0;
		area=length*width;

		System.out.println("Rectangle Area is: "+area);

	}
}
class Demo{
	public static void main(String args[]){
		Rectangle r1=new Rectangle();
		r1.printPerimeter();
		r1.printArea();
		r1.setLength(10.0);
		r1.setWidth(5.0);

		System.out.println("Rectangle length: "+r1.getLength());
		System.out.println("Rectangle width: "+r1.getWidth());

	}
}
```

output,

```
Rectangle Perimeter is: 30.0
Rectangle Area is: 50.0
Rectangle length: 10.0
Rectangle width: 5.0
```

31)

A. Compile Error at line 1
B. Compile Error at line 3

32)
G. None of the above

output,

```
1 200 10 200 100 200
```

33)
line 5 and line 8 are illegal.

34)
Explain difference between “static variables” and “instance variables”

static variables-

- **Static variables are declared using the "static" keyword within the class definition.**
- **They are typically used to store data that is common to all instances of the class and doesn't vary between different objects of the class.**

ex:-

```
class Employee {
int id;
String name;
double salary;
static String ceoName;
}
```

instance variable-

- **Also known as "non-static variables" or "member variables," instance variables are associated with individual instances (objects) of the class.**
- **Instance variables are declared within the class definition but outside of any method or constructor, making them accessible to all methods within the class.**
- **They are typically used to store data that varies between different objects of the class.**

35)

A. Compile Error at line 1
D. Compile Error at line 4
F. Compile Error at line 6
I. Compile Error at line 9

36)

37)
A. Prints: [a,0],[b,1],1,0,0,2

38)
C. Prints: 1,2,3,4,5,3

39)
Explain difference between “static methods ” and “instance methods”

40)
Write a java class called “Cylinder” with attributes length (type double) and radius (type double) of the Cylinder.
a. Create methods that calculate the volume and area of the Cylinder.
b. Implements two constructors default constructor and parameterized constructor.
c. Implements setters and getters of each attributes of the class cylinder.

41)
output,
Box is loaded into memory

42)
output,
A box object is created..
A box object is created..
the “System.out” is printed is an instance block ,so when a new object is initiated,it is printed.

43)
output,
Box is loaded into memory
A box object is created..
A box object is created..
A box object is created..

the “System.out” is printed in an instance block,So when a new object is initiated,it is printed.

static( ){

}
static block is printed only once.

44)
A. 1 2 3

B. 1 2 3 2 3

C. 1 4

D. compile error

E. 1 4

F. no output

45)
B. Compile error at Line 3
D. Compile error at Line 4

46)
Describe “Constructor overloading” and “Method overloading” with examples.

- **constructor overloading -**

Constructor overloading is the process of defining multiple constructors within a class, each with a different number or type of parameters.

ex:

```
class Student {
String id;
double prfMark;
double dbmMark;
boolean status;
Student() {
System.out.println("Student()");
}
Student(int i) {
System.out.println("Student(int i)");
}
}

class Main {
public static void main(String args[]) {
new Student(); //Initiate a new Student Object
new Student(10); //Initiate a new Student Object
}
}
```

- **Method Overloading:**

Method overloading is the process of defining multiple methods within a class, all having the same name but differing in the number or type of their parameters.

ex:

```
import java.util.*;
class Main {
public static void printNumber(int x) {
int rand = new Random().nextInt(101);
System.out.println("Random Number: " + rand);
}
public static void printNumber() {
System.out.println("Number: " + 10);
}
public static void main(String args[]){
printNumber();
printNumber(10);
}
}
```

47)
K. myMethod(x, d);
The correct answer is option K: myMethod(x, d);. It will allow the code to compile because the method parameters match the types of the arguments being passed

48)
valid options,
A. public static void myMethod(char x,double y){}
B. public static void myMethod(double x,char y){}
C. public static void myMethod(int a,double b){}
E. public static void myMethod(byte x,double y){}
F. public static void myMethod(short x,double y){}

49)

C. (5, 3) (3, 5) (5, 3)

50)
Describe “Method Call by Values” vs “Method Call by Reference” with appropriate examples

Method Call by Values:
In a method call by values, the method receives a copy of the actual argument's value. Any changes made to the parameter inside the method do not affect the original value of the argument.

Method Call by Reference:
In a method call by reference, the method receives a reference to the memory location of the actual argument. Any changes made to the parameter inside the method directly affect the original value of the argument.

51)
line 3,line 5 illegal

52)
```
class Cat{
        private String name;
        Cat(String name){
                this.name=name;
        }
        public String getName(){
                return name;
        }
        public void setName(String name){this.name=name;}
}
class main{
        public static void main(String args[]){
        Cat[] cats={new Cat("Aldo"),
        new Cat("Bear"),
        new Cat("Toby"),
        new Cat("Teddy"),
        new Cat("Henry")};
        String catNames[]=new String[cats.length];

        for(int i=0;i<cats.length;i++){
                catNames[i]=cats[i].getName();
        }
        System.out.print("[");
   System.out.print(String.join(", ", catNames));
   System.out.println("]");
        }
}
```

53)
```
class Employee {
   private String firstName;
   private String lastName;
   private double monthlySalary;

   public Employee(String firstName, String lastName, double monthlySalary) {
      this.firstName = firstName;
      this.lastName = lastName;
```

```
        if (monthlySalary > 0) {
            this.monthlySalary = monthlySalary;
        } else {
            this.monthlySalary = 0; // Default to 0 if the salary is not positive
        }
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public double getMonthlySalary() {
        return monthlySalary;
    }

    public void setMonthlySalary(double monthlySalary) {
        if (monthlySalary > 0) {
            this.monthlySalary = monthlySalary;
        }
    }

    public double getYearlySalary() {
        return monthlySalary * 12;
    }

    public void giveRaise() {
        double raiseAmount = monthlySalary * 0.10;
        monthlySalary += raiseAmount;
    }

}

public class Demo {
    public static void main(String[] args) {
        Employee employee1 = new Employee("John", "Doe", 5000);
        Employee employee2 = new Employee("Jane", "Smith", 6000);

        System.out.println("Yearly Salary for " + employee1.getFirstName() + " " +
employee1.getLastName() + ": " + employee1.getYearlySalary());
        System.out.println("Yearly Salary for " + employee2.getFirstName() + " " +
employee2.getLastName() + ": " + employee2.getYearlySalary());

        employee1.giveRaise();
        employee2.giveRaise();
        System.out.println("Yearly Salary after raise for " +
employee1.getFirstName() + " " + employee1.getLastName() + ": " +
employee1.getYearlySalary());
        System.out.println("Yearly Salary after raise for " +
employee2.getFirstName() + " " + employee2.getLastName() + ": " +
employee2.getYearlySalary());
    }
}
```

54)
output,

```
code : 1001
code : 1001
```

55)

56)
C. 10,0,20

57)

58)

```
class Box {
    private int length;
    private int width;
    private int height;


    public Box() {
        length = 0;
        width = 0;
        height = 0;
    }


    public Box(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }


    public Box(int sideLength) {
        this.length = sideLength;
        this.width = sideLength;
        this.height = sideLength;
    }


    public Box(Box otherBox) {
        this.length = otherBox.length;
        this.width = otherBox.width;
        this.height = otherBox.height;
    }
    public void setLength(int length) {
        this.length = length;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public void setDimension(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    public void setDimension(int sideLength) {
        this.length = sideLength;
        this.width = sideLength;
        this.height = sideLength;
    }

    public void setDimension(Box otherBox) {
        this.length = otherBox.length;
        this.width = otherBox.width;
        this.height = otherBox.height;
    }


    public int getLength() {
        return length;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }


    public int getVolume() {
        return length * width * height;
    }


    public void printVolume() {
        System.out.println("Volume: " + getVolume());
    }


    public Box getCopy() {
        return new Box(this);
    }
}
```

```
class Demo {
    public static void main(String args[]) {

 Box b1 = new Box();
        b1.setLength(12);
        b1.setWidth(5);
        b1.setHeight(3);
        b1.printVolume();
        b1.setDimension(120, 50, 30);
        System.out.println("Volume: " + b1.getVolume());

        Box b2 = new Box(4, 2, 3);
        b2.printVolume();

        Box b3 = new Box(b2);
        b3.printVolume();

        Box b4 = new Box(10); // length for a square cube
        b4.printVolume();

        Box b5 = new Box();
        b5.setDimension(12); // length for a square cube
        b5.printVolume();

        Box b6 = new Box();
        b6.printVolume();
        b6.setDimension(b1);
        b6.printVolume();

        Box b7 = b3.getCopy();
        b7.printVolume();
    }
}
```

59)
C. myMethod(b, s);
D. myMethod(s, b);
E. myMethod(x, y);
M.myMethod(x,ch);

60)
G. Compiler error

61)
MyClass()
MyClass(int,int)
MyClass(int)

62)

F. this(i)
D. this(100);