**ENCAPSULATION ASSIGNMENT**

**QUES-1)** WHAT IS ENCAPSULATION IN JAVA? WHY IT IS CALLED DATA HIDING.

**ANS)** ENCAPSULATION - THE WRAPPING OF  DATA MEMBERS AND DATA FUNCTIONS IN A SINGLE UNIT.

BECAUSE VIA USING ENCAPSULATION WE CAN HIDE THE ESSENTIAL DETAILS AND ONLY DISPLAY THE RECOMMENDED FEATURES TO CUSTOMER.

**QUES-2)** THE IMPORTANT FEATURES OF ENCAPSULATION?

**ANS)**

1) DATA HIDING.

2) PROVIDE SECURITY.

3) IT IS EASY AND FAST

4) WE CAN CREATE READ ONLY CLASS AS WELL AS WRITE ONLY CLASS.

**QUES-3)** WHAT ARE GETTER AND SETTERS IN JAVA, EXPLAIN WITH EXAMPLE.

**ANS)**

GETTERS- THE FUNCTION  USED TO GET THE VALUE FROM A FUNCTION, IT RETURN A VALUE, HAVE A RETURN TYPE

SETTERS- THE FUNCTION USED TO SET THE VALUE TO A FUNCTION VIA USING THIS KEYWORD, IT HAS VOID RETURN TYPE.

CONDITION- IF WE ARE USING GETTER AND SETTERS THAN WE HAVE TO MAKE THE DATA MEMBERS PRIVATE, SO IT LEADS TO SECURITY.

FOR EG-

package Encapsulation;

public class Student {

    private int age;

    private int weight;

    //using getters and setters

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public int getWeight() {

        return weight;

    }

    public void setWeight(int weight) {

        this.weight = weight;

    }

    public static void main(String[] args){

        Student ob = new Student();

        ob.setAge(21);

        ob.setWeight(80);

        int stud1age = ob.getAge();

        int stud1Weight = ob.getWeight();

        System.out.println(stud1age);

        System.out.println(stud1Weight);

        Student obj1 = new Student();

        obj1.setAge(22);

        obj1.setWeight(60);

        int stud2age = obj1.getAge();

        int stud2Weight = obj1.getWeight();

        System.out.println(stud2age);

        System.out.println(stud2Weight);

    }

}

**QUES-4)** WHAT IS THE USE OF THIS KEYWORD, EXPLAIN WITH THE HELP OF EXAMPLE.

**ANS)** THIS KEYWORD - IT ALWAYS POINT TO CURRENT OBJECT, IT IS USED TO SET THE VALUE WHERE THE INSTANCE VARIABLE AND LOCAL VARIABLE NAME IS SAME, HELP TO AVOID AMBIGUITY.

FOR EG -

**public class Test4 {**

**// in this we gonna discuss why we need this, if we create a instance variable of samne name and meanwhile pass the same name to a function than the compiler confuses and it creates ambiguity.**

**public static void main(String[] args){**

**prashant p = new prashant();**

**p.show(10);**

**p.display();**

**}**

**}**

**class prashant{**

**int x;**

**void show(int x){**

**this.x = x; // if we don't use the this keyword than this will throw the initialized value of instance variable that is 0.**

**// after using this it will display the 10 value.**

**}**

**void display(){**

**System.out.println(x);**

**}**

**}**

**QUES-5)** WHAT ARE THE ADVANTAGE OF ENCAPSULATION?

**ANS)**

1. IT IS USED TO PROVIDE SECURITY.
2. WE CAN ADD LAYER TO GETTER AND SETTERS TO ENSURE SECURITY.
3. WE CAN CREATE READ ONLY FILE AS WELL AS WRITE ONLY FILE.
4. IT IS EASY AND FAST.

**QUES-6)** HOW TO ACHIEVE ENCAPSULATION IN JAVA WITH AN EXAMPLE.

**ANS)** IN THIS I HAVE USED BOTH THE CLASSES THE GET AND THE SET CLASS.

package Encapsulation;

public class Student {

    private int age;

    private int weight;

    //using getters and setters

    public int getAge() {

        return age;

    }

    public void setAge(int age) {

        this.age = age;

    }

    public int getWeight() {

        return weight;

    }

    public void setWeight(int weight) {

        this.weight = weight;

    }

    public static void main(String[] args){

        Student ob = new Student();

        ob.setAge(21);

        ob.setWeight(80);

        int stud1age = ob.getAge();

        int stud1Weight = ob.getWeight();

        System.out.println(stud1age);

        System.out.println(stud1Weight);

        Student obj1 = new Student();

        obj1.setAge(22);

        obj1.setWeight(60);

        int stud2age = obj1.getAge();

        int stud2Weight = obj1.getWeight();

        System.out.println(stud2age);

        System.out.println(stud2Weight);

    }

}