Page No. : ....
Date : ....../....../......

ASSIGNMENT - 1

NAME : AMRITANSHU KESHARI
BRANCH : COMPUTER SCIENCE ENGG.
ROLL NUMBER : 19402060006
SUBJECT : OOPS
SESSION : 2019 - 22
SEMESTER : 3rd SEMESTER
COLLEGE : GOVERNMENT POLYTECHNIC COLLEGE
ADITYAPUR

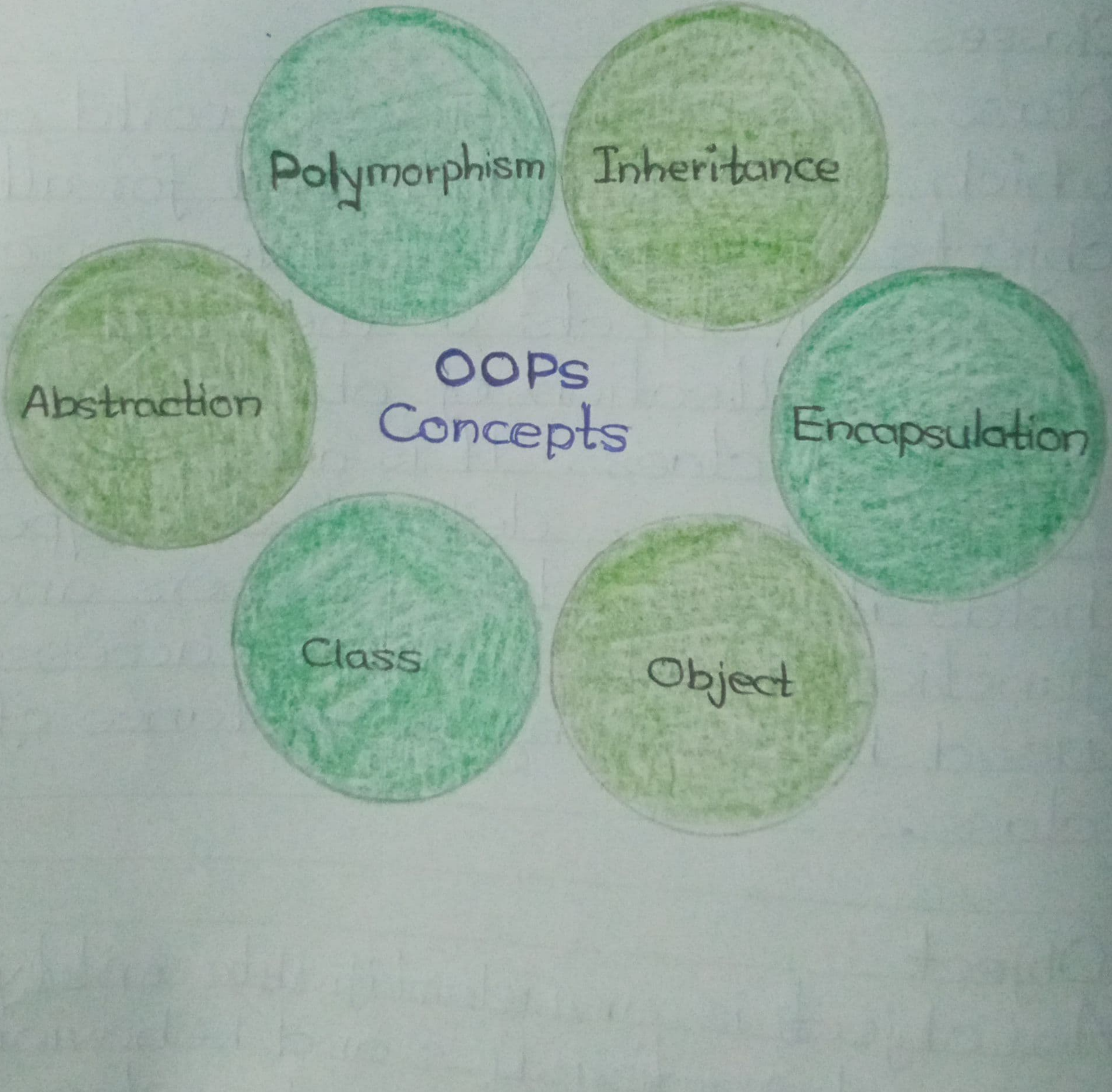
CHACHA BOOK

Features of OOPs :

There are six features of Object Oriented Programming System are as follows as :

1. Classes
Class represents a real world entity which acts as a blueprint for all the objects in a program. We can create as many objects as we need using class. Collection of objects in a program is called class. It is a logical entity. It is a user - defined datatype, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.

2. Object
An object is an identifiable entity with some characteristics and behaviour. An object is an instance of a class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated. Any entity that has state and behavior is known as an


Polymorphism
Inheritance
Abstraction
OOPs Concepts
Encapsulation
Class
Object

object. For example : chair, pen, table, keyboard, bike, computer etc. It can be physical and logical. Object oriented programming system (OOPS) is designed based on the concept of "object". It contains both variables that is used for holding the data and methods that is used for defining the behaviors. We can create any number of objects using this class or a single user defined class and all those objects will get the same fields and behavior. We can set the value for each field of an object.

3. Abstraction

Abstraction is a process where you show only "relevant" data and "hide" unnecessary details of an object from the user. Hidding internal details and showing functionality is known as abstraction. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

Here is the example of abstraction in Object Oriented Programming System (OOPs) as follows as:
=> When you login to your bank account online, you enter your user id and password and press on the login button, what happens when you press login, how the input data sent to server, how it gets verified is all abstracted away from the you.

We can implement the concept of abstraction using classes. The class helps us to group data members and member functions using available access specifiers. A class can decide which data member will be visible to the outside world and which is not.

4. Polymorphism
The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form. When one task is performed by different ways

i.e. known as polymorphism. It is the concept of Object Oriented Programming Systems (OOPS) where an object behaves differently in different situations. Since the object takes multiple forms. For example: A person at the same time can have different characteristics. Like a man at the same time is a father, a husband. an employee. A woman at the same time can have different characteristics are as a mother, a house wife, a teacher and an employee. So the same person posses different behaviour in different situations. This is called polymorphism.

5. Inheritance
When one object acquires all the properties and behaviours of parent object i.e. known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism. The capability of a class to derive properties properties and characteristics from another class is called (Sub class or Derived class which

comes under the inheritance) is called inheritance. Inheritance is one of the most important features of object oriented programming. One class inherits or acquires the properties of another class. Inheritance provides the idea of reusability of code and each sub class defines only those features that are unique to it, rest of the features can be inherited from the parent class.
For example : dog, cat, lion can be derived class of Animal Base Class.

6. Encapsulation
The process of binding or wrapping code and data together into a single unit is known as encapsulation. In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit. In object oriented programming, encapsulation is defined as binding data together and the member functions that manipulate them. Encapsulation also leads to data abstraction or

hiding. As using encapsulation also hides the data. In encapsulation, we wraps both fields and methods in a class, it will be secured from the outside access. We can restrict the access to the members of a class using access modifiers such as private, protected and public keywords. For example: capsule, it is wrapped with different medicines.

Friend Function

A friend function of a class is defined outside that class 's scope but it has the right to access all private and protecte members of the class. Even through the prototype for friend functions appear in the class definition, friends are not member functions.

A friend can be a function, function template or member function, or a class or class template, in which case the entire class and all of its members are friends.

Similarly, protected members can only be accessed by derived classes and are inaccessible from outside. However, there is a feature in C++ called friend functions that break this rule and allow us to access member functions from outside the class. A friend function can access the private and protected data of a class. We declare a frien function using the friend keyword inside the body of the class. It is sometimes useful to allow a particular

to access private members of
other class. Friend function is
like friend class. a friend function
can be given a special grant to
access private and protected
members. A friend function can
be a member of another class.
A friend function can be a
global function in a program.

```
// Program of friend function

#include <iostream.h>
# include <conio.h>

class A {
        private :
            int a, b ;
        public :
            void get_data();
            friend void display (A x);
          };

void A :: get_data ()
      {
        cout << "Enter first number : "<<endl;
        cin >> a ;
        cout << "Enter second number:"<< endl;
        cin >> b ;
      }

void display (A x)
      {
       cout << " You have entered " <<
       x.a << " and "<< x.b << endl;
      }
```

OUTPUT

```
Enter first number:
83
Enter second number:
89
You have entered 83 and 89.
```

```
void main ()
 {
     clrscr();
     A a1;
     a1.get_data();
     display(a1);
     getch();
 }
```