UNIT-1(THEORY SOLUTION)

1. **Explain the notation "Everything is an object" in an object oriented Programming. [PU:2017 spring]**

   An entity that has state and behavior is known as an object. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior. If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running.
   In object-oriented programming problem is analyzed in terms of object and nature of communication between them. Program object should be chosen such that they match closely with real-world objects. In object-oriented programming object's state is stored as data member and behavior is defined using functions (methods).

   When a program is executed, the object interacts by sending message to one another. For example, if "Customer" and "account" are two objects in a program then the customer object may send a message to the account object requesting for the bank balance. Each object contains data, and code (methods) to manipulate data. Objects can interact without having to know details of each other's data or code. It is sufficient to know the type of message accepted, and the type of response returned by objects.
   So, from above discussion we can say that "Everything is an object" in an object oriented programming.

2. With the help of object-oriented programming, explain how can object oriented programming cope in solving complex problem. [PU:2014 spring][PU:2018 fall]

When the software sizes grows large, due to the interconnections of software components the complexity of software system increases. The interconnections of software components means the dependence of one portion of code on another portion. Due to complexity, it is difficult to understand program in isolation as well it makes difficult to divide a task between a several programs.
The abstraction mechanism is used to manage complexity. The Abstraction is mechanism displays only essential information and hiding the details.
Abstraction is often combined with a division into components. For example, we divided the automobile into the engine and the transmission. Components are carefully chosen so that they can encapsulate certain key features and interact with another component through simple and fixed interface.

The division of components means we can divide large task into smaller problems that can then be worked on more less or less independently of each other. It is a responsibility of a developer of a component to provide a implementation that satisfies the requirement of interface. So, set of procedure written by one programmer can be used by many other programmers without knowing the exact details of implementation. They needed only the necessary interface.

From above discussion we can conclude that, breaking down the system in a way such that component can be reuse in solution of different problems just by knowing only the interfaces which must utilize. The implementation details can be modified in future without affecting the program. So program can also easily maintained. In this way object oriented programming cope in solving complex problem.

3. What influence is an object oriented approach said to have on software system design? What is your own opinion? Justify through example. [PU:2009 fall]

Object oriented approach contributes greater programmer productivity, better quality of software and lesser maintenance cost.
The influence of an object-oriented approach is discussed below with certain characteristic of OOP which justifies that object-oriented approach leads to better software design.

- It is easy to model a real system as programming objects in OOP represents real objects. the objects are processed by their member data and functions. It is easy to analyze the user requirements.
- Complexity that arises due to interconnections of software components can be managed by abstraction. So, set of procedure written by one programmer can be used by many other programmers without knowing the exact details of implementation. They needed only the necessary interface.
- It modularize the programs. Modular programs are easy to develop and can be distributed independently among different programmers.
- Large problems can be reduced to smaller and more manageable problems. It is easy to partition the work in a project based on objects.
- It makes software easier to maintain. Since the design is modular, part of the system can be updated in case of issues without a need to make large-scale changes.
- Elimination of redundant code due to inheritance, that is, we can use the same code in a base class by deriving a new class from it. The reusability feature of OOP lowers the cost of software development.
- Reuse also enables faster development. Object-oriented programming languages come with rich libraries of objects, and code developed during projects is also reusable in future projects.
- It provides a good framework for code libraries where the supplied software components can be easily adapted and modified by the programmer. This is particularly useful for developing graphical user interfaces.
- In OOP, data can be made private to a class such that only member functions of the class can access the data. This principle of data hiding helps the programmer to build a secure program that cannot be invaded by code in other part of the program.
- With the help of polymorphism, the same function or same operator can be used for different purposes. This helps to manage software complexity easily.

4. What is the significance of forming abstractions while designing an object oriented System. Explain with a example. [PU:2015 fall]

Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.
Consider a real life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc in the car.
The significance of forming abstractions while designing an object oriented system can be listed as follows:

- It manage complexity that arises due to interconnections of software components.
- Set of procedure written by one programmer can be used by many other programmers without knowing the exact details of implementation. They needed only the necessary interface.
- Data abstraction increases the reusability of the code by avoiding any chances of redundancy.
- It increases the readability of the code as it eliminates the possibility of displaying the complex working of the code.
- With the implementation of classes and objects, comes enhanced security. Since data abstraction is a method of implementing classes and objects any denying access to other classes of accessing the data members and member functions of the base class.
- Helps the user to avoid writing the low level code.
- It separates the entire program into code and implementation making it more comprehensible.
- Can change internal implementation of class independently without affecting the user level code.
- Helps to increase security of an application or program as only important details are provided to the user.
- To increase security of an application or program as only important details are provided to the user.

5.In your opinion how does the object-oriented thinking solve the software crisis? List the basic characteristics of object oriented programming and explain any two characteristics that are directly applicable for software maintenance.

(Students are encouraged to study about software crisis first ***Ref: E. Balaguruswamy book)***

In my opinion, object-oriented thinking can help solve the software crisis by providing a more modular and flexible approach to software design, development, and maintenance. Object-oriented programming (OOP) emphasizes the use of objects, classes, inheritance, and polymorphism to create reusable and scalable software components. By breaking down complex systems into smaller, more manageable pieces, OOP can help reduce the complexity and improve the maintainability of software.

Here are some basic characteristics of object-oriented programming:

Abstraction: OOP allows developers to create abstract representations of real-world objects and concepts, and to focus on the essential features and behaviors of those objects. This can help simplify the design and implementation of software systems.

Encapsulation: OOP allows developers to encapsulate data and behavior within objects, and to control access to those objects through well-defined interfaces. This can help improve the security, reliability, and maintainability of software systems.

Inheritance: OOP allows developers to define new classes based on existing ones, and to inherit and reuse their properties and methods. This can help reduce code duplication and improve code reuse.

Polymorphism: OOP allows developers to create multiple implementations of the same interface or method, and to choose the appropriate implementation at runtime based on the type of object being used. This can help make software systems more flexible and extensible.

Two characteristics of OOP that are directly applicable for software maintenance are inheritance and polymorphism.
Inheritance: This allows developers to create new classes based on existing ones, and to reuse and extend the properties and methods of those classes. This can help reduce code duplication and improve code reuse, which in turn can help simplify maintenance tasks. For example, if a bug is found in a base class, fixing it in the base class can automatically fix it in all the derived classes that use it. Similarly, if a new feature is added to a base class, it can be automatically available to all the derived classes. This can help reduce the amount of code that needs to be maintained and tested, and can also help improve the consistency and correctness of the code.

Polymorphism: This allows developers to create multiple implementations of the same interface or method, and to choose the appropriate implementation at runtime based on the type of object being used. This can help make software systems more flexible and extensible. For example, if a new type of object is added to the system, it can be integrated seamlessly with existing code as long as it conforms to the same interface or method. This can help simplify maintenance tasks because changes can be made without affecting other parts of the program that rely on the interface or method.