Question 3 Explain With Coding:

|  |  |
| --- | --- |
| -Compile Time | Runtime Ploymorphism |
| it is the time at which the source code is changed over into an executable code while the run time is the time at which the executable code is begun running.  In code:   var a=20;  console.log("The value of b is : ",b): | The runtime mistakes are done during execution and after The examples of runtime errors are division by zero, etc. These errors are not easy to detect as the compiler does not point to these errors.  In code:  var a=20;  var b=a/0;  console.log("The value of b is :",b): |

====================================

|  |  |
| --- | --- |
| - Abstraction | Encapsulation |
| It is used for hiding implementation details  In code  class Shape {      constructor (){   var area;   var circumfrence;    void displayResult;      }     class Rectangle extends Shape{       length;       width;        constructor(length, width){            this.length=length;            this.width=width;        }        area(){            const rectArea= this.length\*this.width;            return rectArea;        }      circumfrence(){            const rectCircum= (this.length+this.width)\*2;            return rectCircum;          }        displayResult()        console.log(RectangleArea =${this.area});        console.log(RectangleCircumfrence =${this.circumfrence});      }  } | This concept is also often used to hide the internal representation, or state, of an object from the outside.  In code :   class Person{         name;      idNum;    age;    constructor(name,idnum,age){        this.name=name;        this.idnum=idnum;        this.age=age;    }    getName(){      return this.name;         }         getAge() {         return this.age;      }        getIdNum(idnum) {        return this.idNum;      }     setAge( newAge) {         age = newAge;      }     setName( name) {         name = newName;      }    setidnum(  idnum) {         idnum = idnum;      }   }  console.log("Name : " +getName() + " Age : " +getAge()+ getIdNum()); |

====================================

|  |  |
| --- | --- |
| Static Members | Instance Members |
| **They are class members that are declared using the static keyword. There is only one copy of the static data member in the class, even if there are many class objects. The static data member that is void no return function.**  **In code :**  class Student {     name;      marks;   objectCount;      Student() {         objectCount++;      }         getdata() {        console.log=prompt("Enter name:");        console.log=prompt("Enter marks");        }         putdata() {          console.log=prompt("Name is");          console.log=prompt("Mark is ");      }   }; | **instance** members are defined in a class , it is a variable defined in a class, but is non-static and it return a value.  In code:  class Number{   x;  function() {          return x;      }    setX(newX) {          x = newX;      }; |

-====================================