DT Big Data Exercise

Conceptual questions

1. What is Object Oriented Programming?

- Object oriented programming is a programming methodology in which we deal with objects. These objects are defined in the form of classes. These classes contain data or fields, known as variables and code to operate on these variables known as methods. Objects can be used to represent real life entities within a program. For e.g. consider entity student, which will have attributes such as student id, name, contact no etc. These attributes or properties could be stored in variables and then we can define methods to perform various operations like input data, calculate average, update data etc. Object oriented programming languages support features such as encapsulation, inheritance, polymorphism etc. Some popular object oriented programming languages are java, c++, c#, python etc.

2. What is access Specifier?

- Access specifiers or access modifiers are specifications for methods and variables which define where can these members of the class or interface can be accessed from. Access specifiers enable us to implement the encapsulation feature of object oriented programming. They help us to control accessibility of object members outside the parent class or interface. Java supports 4 type of access specifiers namely:

i. Public: Variables, methods or constructors defined as public can be accessed by any java program whether these classes are within the same package or not.

ii. Private: This is the most restrictive access specifier. Fields, methods or constructors defined as private can only be accessed within their parent class.

iii. Protected: Class members declared as protected can only be accessed by the other members of the same package and also by their subclasses which could be present in another package;

iv. Default: If no access modifier is mentioned while declaring class members then it follows default accessibility level. There is no default access specifier keyword. Using default specifier, we can use fields or methods within same package only.

3. What is the Static and why it is required?

- Static keyword can be used for defining class variables or methods. Variables define static hold values for class and not an instance of it. Hence variables which are to be initialized only once can be declared using static keyword. If a variable is declared is as static it would be initialized when the class is first loaded even if no object for that class is created. Hence memory is allocated only once and all instances of that class will get the same value for that variable.

4. What is Array?

- An array in java is a data structure which can be used to store multiple values of the same datatype. The size of these arrays needs to be declared before using them because of which we cannot increase or decrease the array size at runtime. The members of the array can be accessed using indexes, first element is stored with an index 0. Hence the indexes for an array would be from 0 to n-1 where n is the total number of elements or size of the array. Arrays could be single or multidimensional.

5. What is Data Type?

- Data type is the kind of data item a variable can take. Depending on the need in the program we can choose an appropriate data type for the variable. For e.g. if I want to store integer or numeric value then we would have to declare the variable as numeric or integer type. Data types in java are classified as primitive and non-primitive datatypes.

Primitive Data types:

i. Int
ii. Long
iii. Float
iv. Double
v. Char
vi. Byte
vii. Short
viii. Boolean

Non-Primitive Data types:

i. String
ii. Array etc.

6. What is variable?

- Variables are user defined names for an area reserved memory locations. Variables can be considered as labels for a memory location. It can be assigned value during compiling or can be updated at runtime. Variables are mainly classified into three types:

i. Instance variable: Instance variables belong to a particular instance of the class. Hence the same variable can hold different values for different instances of the class. Every time a new instance of the class is created a new memory location is allocated for the variable for that particular instance. It can be used to store values which would usually be different for different instances of the class for example in a class that stores student details, the details of students such as name, id, contact details which would hold unique values for different students will be defined as instance variables.
ii. Static variables: Static variables hold the same value all instances of the same class. Hence they are also called as class variables. Memory is allocated to them only once

that is during class creation. In the example discussed above static variables could be used to store values that could be common to all the students like college name.

iii. Local variables: Local variables are the variables that are declared inside the method. They exist only within the method and their scope is limited till the closing braces of the method.

7. What is operator and why it is required in programming?

- Operators are special symbols which can be used to perform specific operations on one or more operands and then return the result. Some of the operators in java are:

i. Arithmetic operators: These are the operators used to perform arithmetic operations. E.g. Addition (+), Subtraction (-), Multiplication (*), Division (/) and Modulo (%).

ii. Unary Operators: Unary operands only require single operand. They are used to perform increment or decrement operations. E.g. Increment operator (++) and decrement operator (--).

iii. Assignment operator: Used to assign values to the variables. E.g. =, +=, -=, /=, *=, %=

iv. Relational operator: Used to check relations like greater than (>), less than (<), equal to (==), not equal to (!=), less than or equal to (<=) and greater than or equal to (>=).

v. Logical operators: Used to perform logical ‘and (&&)’ or ‘or (||)’ operations.

vi. Bitwise operator: Used to perform bit manipulation operations. E.g. Bitwise and (&), bitwise or (|), bitwise exor (^) and bitwise complement (~).

vii. Shift operator: These operators are used to shift the bits of a number left or right thereby multiplying or dividing the number by two respectively. They can be used when we have to multiply or divide a number by two.

viii. Ternary o perator: It is a shorter version of the if else condition. It takes three operands.

8. What is conditional constructs and why it is required in programming?

- Conditional constructs are used to check for certain conditions and depending on the result of this comparison to take certain predefined step and control the code execution. Java supports if-else and switch case as conditional statements.

9. What is difference between If--else if and switch case?

- If-else and switch are both conditional statements i.e. they both change the flow of the program depending on whether the condition is true or false. Expression inside if statement decide whether to execute the statements inside if block or under else block. On the other hand, expression inside switch statement decide which case to execute. If-else statement checks for equality as well as for logical expression whereas switch checks only for equality. We can have multiple if statement for multiple choice of statements. In switch we only have one expression for the multiple choices.

10. What is Looping Constructs and why it is required in programming?

- Looping constructs are used to execute the same block of code again and again, as many times required. This often saves times since we only have to write the repeating code only once. Mainly three types of loops are supported in java namely:

i. For loop.
ii. While loop
iii. Do—while loop

11. What are the different phases of Looping?

- Different phases of looping are:

i. Initialization phase: In this phase the loop control value is initialized to the starting value.
ii. Condition phase: In this phase we check whether the predefined condition for executing the loop are met or not.
iii. Updating phase: In this phase we update the control variable and proceed for the next iteration of the loop.

12. What is difference between wshile, do--while and for loop?

- While loop and for loop are entry controlled loops, i.e. the loop condition is checked while entering the loop whereas do while loop is an exit controlled loop which means the condition is checked while exiting the loop. Hence in case of while or for loop if the loop condition doesn't hold true for the first iteration the loop would not be executed even once however in case of do-while loop the loop is executed at least once on matter the loop condition stands true or not.

- In case of for loop the initialization and updating loop variable is the part of the syntax whereas it is not the case with the while and do-while loop.

13. What is Array?

- An array in java is a data structure which can be used to store multiple values of the same datatype. The size of these arrays needs to be declared before using them because of which we cannot increase or decrease the array size at runtime. The members of the array can be accessed using indexes, first element is stored with an index 0. Hence the indexes for an array would be from 0 to n-1 where n is the total number of elements or size of the array. Arrays could be single or multidimensional.

14. What Is Array in Java?

- An array is a group of like-typed variables that are referred to by a common name. In Java all arrays are dynamically allocated. The variables in the array are ordered and each have an index beginning from 0. Java array can also be used as a static field, a local variable or a

method parameter. The size of an array must be specified by an int value and not long or short.

- Syntax for declaring a one dimensional array in java:

    Datatype[] array-name;

    To initialize the array:

        Array-name = new datatype[array-size];

    e.g. Declaration: int[] id;

    initialization id = new int[10];

- Similarly, for two-dimensional array:

    Declaration: int[][] matrix;

    Initialization: matrix = new int[3][3];

15. What is function return type?

- A function return type is the datatype of the variable that the function or method will return on completing its execution. The function return type has to be mentioned with the function declaration. If in case the function is not going to return any variable to the calling function, then the return type is void and void keyword has to used before the function name.

E.g. int add() – here we are declaring the return type of add() function as int and hence jvm would be expecting the add function to return a variable of type int.

16. What is the use of passing parameter inside the function?

- In object oriented programming we can use separate methods or functions to perform separate operations, this helps us to implement code reuse which means we only have to write the code for the function only once and then we can call the function as and when required.

- However in certain cases we might have to pass some values to the function on which we might have some operations to be performed by the function. To pass these values we would have to pass them as function parameters.

- These parameters would have to mentioned within the function declaration and be passed when calling the function.

E.g. void add(int a, int b){

    System.out.println(a+“ + ”+b+ “ = ”+(a+b));

}

Function call: add(2,3);

- In the above example we created a function add() which performs the addition operation on two integer values passed to it and displays the result. These values have to be passed with the function call as we did with add(2, 3) command.

17. What is void and why it is used?

- Void is a datatype which is used with the method declaration which says that the method is not going to be returning any variable after execution. Whenever the method is called it would just execute its code and just pass the control to the calling function without returning any value to it.

18. What is use of String[] aa in main function?

- String[] variable-name is used in the main function to input command line arguments. These arguments can be passed to the main function while starting the execution of the main function. We can take any number of arguments through the command line and these values would be stored in an array of datatype string, then depending on the type of data we required we can typecast the values from the string array to that type.

19. What is the difference between case condition of Java and C++?

- The major difference in the case condition in java and c++ is that in java we could place variables in the case condition which would be calculated at runtime and hence can be dynamically changed however in case of c++ the case statement can only hold constant values.

20. What is Exception Handling?

- Whenever an error occurs during program execution, the execution stops abruptly and an exception is thrown. However, in order to ensure that the program completes execution, whether or not error occurs, we can use certain techniques to handle the error and continue the program execution. This process is known as exception handling.

21. Why do we require Exception Handling?

- The main need of exception handling is to maintain the normal flow of the program. Whenever an exception occurs it interrupts the normal flow of the program and the code after the exception causing statements would never be executed. To make sure that the rest of the code is executed and the program completes its execution we use exception handling.

22. How Exception Handling can be implemented in the program/How to handle exception in program?

- When an error occurs within a method, the method creates an object and hands it off to the runtime system. The object, called an exception object, contains information about the error, including its type and the state of the program when the error occurred. Creating an exception object and handing it to the runtime system is called throwing an exception.

- The runtime system searches the call stack for a method that contains a block of code that can handle the exception. This block of code is called an exception handler. The search begins with the method in which the error occurred and proceeds through the call stack in the reverse order in which the methods were called. When an appropriate handler is found, the runtime system passes the exception to the handler. An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.

-The exception handler chosen is said to catch the exception, mentioned steps are taken and the program is terminated.

E.g.
```
try {
    //Block of code that can throw exception
}
Catch (exception e) {
    //Exception handling code
}
Finally {
    // Finally block
}
```

- Finally block is executed whether or not exception occurs. Finally block is usually used for cleaning up resources such as closing files, connections, etc.

23. What is constructor?

- Constructors are like methods except the fact that they don't have a return type. Constructors are always executed whenever a new instance of the class is created. They are usually used to initialize the class variables. A constructor must have the same name as the class. Depending on the requirement the constructor could be parameterized or no argument constructor also known as default constructor.

24. What is the use of constructor?

- A constructor is usually used to initialize the values of class variable. Since a constructor is executed by default every time a new instance of the class is created we can use the constructor to initialize the variables for that instance.

25. What is parameterized constructor?

- A constructor that has parameters in it is called as parameterized constructor, it is used to assign different values for different objects. For e.g. Consider a class that takes student details.

```
class studentDetails{

        Int id;

        String name;

        public studentDetails(int id, String name) {

                this.name = name;

                this.id = id;

        }


        public static void main(String[] args){
                studentDetails s = new studentDetails(101, "Tom");

                System.out.println("Student Details: "+s.id+ " "+s.name);

        }

}
```

- In the above example we create a class studentDetails, constructor studentDetails that initializes name and id fields when the object s is created for the class.

26. What are the rules to be followed while using constructor?

- Rules to be followed while using constructor:

    i. The class constructor should have same name as the class it belongs to.

    ii. The constructor doesn't have a return type.

    iii. Constructor cannot be abstract, final, static or Synchronized.

    iv. Access specifiers can be used while declaring the constructor so as to control which classes can access the constructor.

27. How do we use the base class constructor in child class?

- In case the base class only has a default constructor than it is automatically called when the object of the child class is created.

- However in case of a parameterized constructor, we need to use super keyword along with the list of arguments to call the base class constructor.

28. What will be the flow of constructor execution while implementing inheritance?

- The flow of constructor execution while implementing inheritance goes from the base or parent class to its child class.

- Hence whenever an object of the child class is created, the constructor of its parent class are executed and then the constructor if the child class will be executed implicitly.

29. What is function overriding? Why it is required? How it can be implemented in program?

- Having a function with the same definition in the child class as one already present in the parent class is called as function overriding. Using function overriding the child class can have its own implementation of the function, hence implementing a more specific version of the same function but as per the requirement of the child class.

- For e.g. consider a parent class vehicle, having child class car. Now vehicle would some generic attributes that would be present for all its child classes, like color, company, number of passengers it can carry etc. However, in case of the class car there could be certain values that would hold true for car and not for other vehicles. These functions in the child class could override the functions in the parent class without having to modify the code of the parent class.

- Rules for implementing function overriding:

    i. Function must have the same name as the one in the base class.

    ii. Function must have the same parameter list.

30. What is function Overloading? Why it is required? How it can be implemented in program?

- Function overloading is having more than one functions in a class with the same return type and function name but different parameter list. Now the parameter list could differ in terms of number of parameters, their datatypes or the order in which they are passed.

E.g.

```
Class Addition{

    void add(int a, int b){

        System.out.println(a+b);

    }

    Void add(float a, float b){
        System.out.println(a+b);

    }

}
```

-In the above example the Addition has two methods both with the same return type and having the same name add(). However, one takes integer parameters and the takes float values as parameters and hence depending on what type of parameters are passes in the function call the corresponding function would be executed.

31. What is Inheritance and how it can be implemented in program?

- Inheritance is a feature of object oriented programming through which one object can acquire the functions and variables of another function. The function whose features are acquired is known as parent or base class and the class that acquires these features is known as the child class. It helps in achieving code reusability as we only need to write the code for general functions in the base class and the child classes can just inherit these functions and then would only need to implement the class specific functions.

```
E.g. class Teacher {
        String designation = "Teacher";
        String collegeName = "Abc";
        }

public class PhysicsTeacher extends Teacher {
        String mainSubject = "Physics";
        public static void main (String args[]) {
                PhysicsTeacher obj = new PhysicsTeacher();
                System.out.println(obj.collegeName);
                System.out.println(obj.designation);
                System.out.println(obj.mainSubject);
        }
}
```

Output:

```
Abc
Teacher
Physics
```

32. How to inherit Class and how to inherit interfaces?

- To inherit class java we need to use the extends keyword. The extends keyword is specified after the class name and before another class name. The class name before extends identifies the child and the class name after extends identifies the parent. It's impossible to specify multiple class names after extends because Java doesn't support class-based multiple inheritance.

```
class Vehicle

{

        //parent class member declaration

}

class Car extends Vehicle

{

        // inherit accessible members from Vehicle

        // provide own member declarations

}
```

- In the above example the Vehicle class is the parent class and the Car class is the child class.

- Interfaces in java are abstract classes i.e. they only contain variables and method signatures. The methods declared inside the interface must be implemented by the inheriting class. The variables declared inside an interface are treated as constants because they are implicitly public, abstract and final. To inherit an interface the inheriting class needs to use the implement keyword and also implement the abstract methods mentioned in the interface.

E.g.

```
        Interface myInterface{

                Public void method1();

        }

        Class myClass implements myInterface{

                Public void method1(){

                        //Implementation of method1

                }
```

```
        Public static void main(String[] args){

                myInterface m = new myClass();

                m.method1();
        }

}
```

- The above example shows how to create an interface and then implement it using another class.

33. What is interface and how it is different from Abstract class?

- An interface is can be called as a blueprint of a class, as in it contains the constants and method signatures that are going to defined in the class.

- An interface and an abstract class both contain abstract methods.

- An abstract class can have non-static and non-final variables but an interface can only have static and final variables which makes them constants.

- Also a class can extend only one abstract class but can implement any number of interfaces.

34. What is framework on Java?

- Framework in java is a platform which contains classes which we can use in our code to achieve our required task. We can make use of framework by calling its functions, inheriting their classes or using other implementations of it.

- For e.g. a framework might contain functions such as taking input from the input devices, classes and functions to perform operations on them, interacting with the system software or displaying output on through the output devices. Hence, frameworks speed up the development process as we do not need to implement these functions ourselves.

35. What is Java Collections?

- Java collections are a set of functions and interface that define some commonly used data structures. Using these functions or interfaces we can make use of these data structures and perform various operations on them without having to write lengthy codes for them.

- Java collections provide an architecture to store data into various data structures and perform various manipulation operations on them.

36. What is collection Framework?

- Collection framework provides us with a set of classes and interfaces which can be used to represent a group of data as a single unit.

- It reduces the development time and also the effort required for learning and implementing various data structures and also helps in implementing software reuse.

37. What is ArrayList?

- Arraylist is part of the java collections framework. It is more flexible than arrays as its size can be dynamically increased or reduced as per requirement. Arraylists allow us to randomly access the list. An arraylist cannot be used with primitive data types like int, float, char we need to use their respective wrapper class to store them in arraylists.

38. What is Iterator?

- An iterator is a control variable which is used to traverse through collection objects such as arraylists, linked lists, hash maps. Iterators help in retrieving the elements of the list one by one.

39. What are the characteristics of ArrayList?

- An ArrayList can be dynamically resized. Depending on how and when we add data into the ArrayList the size keeps on increasing and doesn't have to be set beforehand. It doesn't need to have an upper bound specified.

- An ArrayList can only hold objects and not primitive data types. Hence, to store data of any primitive data type we need to use their respective wrapper classes.

- ArrayList has inbuilt methods such as add(), clear(), indexOf(), remove(), size(), etc. to perform various operations on the ArrayList.

40. What are the advantage of using ArrayList?

- While using an ArrayList we do not necessarily need to define the capacity while declaration. We can keep on adding data and as we perform various operations on the ArrayList the size is automatically adjusted.

- Internally the content is stored in an array which would be having a fixed size but when this array gets full, a new, bigger array is created and used.

- An ArrayList has predefined methods which can be used to perform various operations on them.

41. Application of ArrayList in Real life software?

- An ArrayList can be used to store details such as say contact details in a phonebook. Now the number of contacts a user might have cannot be predicted or fixed and hence using an ArrayList will help in managing these contacts in a more efficient manner.

- Another example where ArrayList can be used are process queues. Here processes can come in any number and before they are executed they need to be stored in a buffer. Now this buffer cannot have a fixed size and hence dynamic arrays or ArrayList will be perfect in this case.

42. What is LinkedList?

- LinkedList is the java implementation of doubly linked list. As it represents a doubly linked list, it can be traversed in both directions. A LinkedList can contain duplicate elements and it maintains the order in which the elements were inserted. A LinkedList can be traversed using ListIterator.

- Some of the methods supported by LinkedList are add(), addFirst(), addLast(), size(), getFirst(), getLast(), etc.

43. What is ListIterator?

- A ListIterator is an iterator for list objects, i.e. it can be used to retrieve the elements of the list one by one. It can also be used to perform create, read, update and delete operations on the list.

- Unlike Iterator, ListIterator supports both forward and backward iterations.

44. What are the characteristics of LikedList?

- A LinkedList can be traversed in both forward and backward directions.

- It supports the CRUD operations i.e. create, read, update and delete operations.

- A LinkedList is a dynamic data structure i.e. it can increase or decrease in size as per requirement.

- A LinkedList can contain duplicate elements

- LinkedList maintains the order in which data is inserted into it.

45. What are the advantage of using LinkedList?

- Since LinkedList is a dynamic data structure we don't need to define its size while declaring it. Its size increases automatically as we keep on inserting data.

- Also since we don't need to specify the size beforehand, memory is allocated as the size grows and hence the LinkedList doesn't holds any extra space i.e. no memory wastage.

- The methods that are available to operate on LinkedList make implementing data structures such as stack very easy to implement.

46. Application of LinkedList in Real life software?

- Since LinkedList contains links to the previous and the next node, they can be used to store the sequence of operations performed and in case we want to undo these operations or replicate them it can easily be done using LinkedList.

- Implementing data structures such as stack, queues, graphs etc.

47. What is the difference between LinkedList and ArrayList?

- ArrayList is implemented using arrays whereas LinkedList is implemented using doubly linked list.

- ArrayList is slower whereas LinkedList is faster for storing and retrieving data.

- ArrayList is more efficient for storing data, however, if may data manipulation operations are to be performed than linked list works better.

- ArrayLists only store the data and the index whereas LinkedList has to store the pointers to the next and the previous nodes hence there its requires more memory.

48. What is generic Class?

- Generic class allows us to us to define common classes for different datatypes and depending on requirement use the required data type.

- We can define our own classes with generics type. A generic type is a class or interface that is parameterized over types.

- It also helps in ensuring compile time safety by helping in catching invalid type errors at compile time.

49. What are the benefits of Generic Class?

- Java Generic methods and generic classes enable programmers to specify, with a single method declaration, a set of related methods, or with a single class declaration, a set of related types, respectively.

-Generics also provide compile-time type safety that allows programmers to catch invalid types at compile time.

-Using Java Generic concept, we might write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

50. Uses of Generic class in real life scenario?

- We can write a generic method for sorting an array of objects, then invoke the generic method with Integer arrays, Double arrays, String arrays and so on, to sort the array elements.

- Another example could be having a common class for vehicles and then depending on what type of vehicle is required at that instance, appropriate functions could be used.

51. What is HashSet?

- A HashSet in java is an implementation of the hash table. A hash table stores data in the form of key-value pair.

- A HashSet doesn’t allows duplicate data and hence if an entry with an already existing key is made it overwrites the previous entry.

- Some of the methods to operate on a HashSet are add(), contains(), clear(), remove(), etc.

52. What is the difference between List and Set?

- A list can store duplicate values however, a set can only store unique values.

- List is an ordered sequence whereas set is not.

- A list provides positional access whereas a set does not.

53. What are the characteristics of HashSet?

- HashSet stores data in the form of key-value pairs.

- HashSet doesn't store duplicate values.

- HashSet is an unordered set, the values aren't stored in the order in which they are added.

54. What are the advantage of using HashSet?

- Insertion and search operations work faster with HashSet.

- Since it doesn't take duplicate values, it can be helpful in maintaining consistency.

55. Application of HashSet in Real life software?

- HashSets can be used to store details such as phonebook where the key could be name or an hash function of it.

- Another example could be a social media website where the key could be an hash function of the user's email id or user id and it can be used to store other user data.

56. Why to Use constructor?

- Constructor is a block of code that initializes the newly created object.

- Since constructors are executed every time an instance of the class is created, constructors can be used to initialize the instance variables.

- Hence every time we create a new instance of the class; we can set the values for the variables of that instance and automatically memory allocation takes place.

57. How to use constructor?

- The constructor name must be same as the class name. Constructors usually do not have a return type.

- The constructor will automatically be executed when a new instance of its class is created.

Syntax for creating a constructor:

```
class myClass{
        public myClass(){

                System.out.println("This is printed from the constructor.");

        }

        public static void main(String[] args){

                myClass obj = new myClass();

        }

}
```

Output:

```
This is printed from the constructor.");
```

58. What is the difference between constructor and Normal Function in Java?

- Constructors need to have a same name as the class name whereas this is not the case with function name.

- A function could have a return type however; a constructor doesn't have a return type.

- A function has to be explicitly executed, however the constructor gets automatically executed whenever an instance of its class is created.

- Constructors are usually used to initialize instance variables whereas functions are used to perform some operations on these variables or expose these variables to the outside world.

59. What is Inheritance?

- Same as question 31.

60. How to implement Inheritance?

- Same as question 32.

61. What is the use of Access Specifier while using Inheritance?

- Depending on the scope of the access specifier the behavior can change while using inheritance.

- Methods or variables declared as public are accessible from anywhere and hence they will be available to the subclasses as well.

- Private members of the class are not accessible outside the class, so if in case we need to use these objects outside the class we will need to pass their values using the getter and the setter methods.

- Protected members are as it is visible to the subclasses and hence can be retrieved from the child classes.

- Default type members are also not directly visible to the subclasses and hence they require similar steps such as the once required by the private members.

62. Why multiple inheritance is not supported in Java?

- Main reason behind java not supporting multiple inheritance is the ambiguity that could arise when inheriting multiple classes.

- For example let's say there are two classes A and B which are inherited by class C. Now if these classes contain methods with the same name and this method is called from inside C then there would be ambiguity as in which method to execute. This problem is called as Diamond Problem.

- However we can achieve multiple inheritance by extending one class and implementing one or more interfaces.

63. How constructors behave wile implementing Inheritance? What is the use of this and super keywords in java?

- When implementing inheritance, the constructors are executed in order from the parent class to the subclasses. The constructor of the parent class can be executed by using super keyword in the constructor of the child class.

- **This** keyword is used in java to refer to the current instance of java. It can be used to invoke the constructor of the current instance of the class. This keyword is basically used to refer to any member of the current object from an instance method or constructor.

- Like this keyword is used to refer to the current object, **super** keyword is used to refer to the variables, methods or constructors of the super class. Only public and protected members of the class can be called using the super keyword.

64. What is polymorphism? Benefits of polymorphism?

- Polymorphism stands for taking many forms.

- Polymorphism in terms of java describes the ability of a language to objects of various types and classes through a single, uniform interface.

- There are two types of polymorphism supported by java namely: compile time polymorphism and runtime polymorphism.

-Method overloading is an example of static polymorphism, while method overriding is an example of dynamic polymorphism.

65. How polymorphism is implemented?

- There are two types of polymorphism in java: compile time polymorphism and runtime polymorphism.

- Runtime polymorphism or Dynamic Method Dispatch is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

- In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

- At compile time, Java knows which method to invoke by checking the method signatures. So, this is called compile time polymorphism or static binding.

66. What is method Overloading? How it can be implemented? What are the rules to be followed while implementing method overloading?

- Method overloading is the process of declaring multiple methods in a class, having the same method name but different parameter lists.

- Depending on the type and number of parameters the corresponding method would be executed.

- The methods that are to be overloaded should have same method name.

- The parameter list should be different for each method either in terms of number of parameters, data type of these parameters or the order in which they are to be sent.

67. What is method Overriding? How it can be implemented? What are the rules to be followed while implementing method overriding?

- Having a method with the same definition in the child class as one already present in the parent class is called as method overriding. Using method overriding the child class can have its own implementation of the method, hence implementing a more specific version of the same method but as per the requirement of the child class.

- For e.g. consider a parent class vehicle, having child class car. Now vehicle would some generic attributes that would be present for all its child classes, like color, company, number of passengers it can carry etc. However, in case of the class car there could be certain values that would hold true for car and not for other vehicles. These functions in the child class could override the functions in the parent class without having to modify the code of the parent class.

- Rules for implementing method overriding:

    i. Method must have the same name as the one in the base class.

    ii. Method must have the same parameter list.

68. What is the use of Final Keyword?

- When a variable is declared with final keyword, it's value can't be modified, essentially, a constant. This also mean that we have to initialize a final field.

- If the final variable is a reference, this means the variable cannot be reassigned to reference another object, but internal state of the object pointed by that reference variable can be changed i.e. we can add or remove elements from final array or final collection.

- The only difference between a final field and a non-final is that a non-final can be reassigned a value but a final field once assigned cannot be changed later.

Programming Questions

Qu 1.

Write a program in Java to accept the details of 10 Students. Display the total count of students who are eligible for taking admission in Graduation 1st Year if age is greater than 18.

Details will be

studentName
StudentAge

```
package javaExercise;
import java.util.*;

public class Qu1 {

	public static void main(String[] args) {
		String name;
		int age;
		Scanner s= new Scanner(System.in);
		ArrayList<Student> a = new ArrayList<Student>();
		for(int i=0; i<3; i++) {
			System.out.println("Enter student name: ");
			name = s.next();
			System.out.println("Enter student age ");
			age=s.nextInt();
			Student stud = new Student(name, age);
			a.add(stud);
		}
		int elig=0;
		for(Student st: a) {
			if(st.sage > 18) {
				elig++;
			}
		}
		System.out.println(elig+" student(s) are eligible for taking admissions
into graduation 1st year.");
	}

}

class Student {
	String sname;
	int sage;
	public Student(String sname, int sage) {
		this.sname = sname;
		this.sage = sage;
	}
}
```

Qu 2.
Write a program in Java to accept the details of 10 Employees.Display the total bonus given to the employees during festival season. Bonus Criteria is given below.

if Salary is . 10000 Bonus will be 30%
if Salary is . 50000 Bonus will be 20%
if Salary is . 100000 Bonus will be 10%
Details will be : EmployeeName : EmployeeSalary

```
package javaExercise;
import java.util.*;
public class Qu2 {

	public static void main(String[] args) {
	Scanner s=new Scanner(System.in);
	String name;
	int sal;
	double total = 0;
	LinkedList<Employee> emp = new LinkedList<Employee>();
	for(int i=0; i<3; i++) {
		System.out.println("Enter employee name: ");
		name = s.next();
		System.out.println("Enter employee salary: ");
		sal=s.nextInt();
		if(sal<=50000) {
			total+=(sal*0.3);
		}
		else if(sal>50000 && sal<=100000) {
			total+=(sal*0.2);
		}
		else if(sal > 100000) {
			total += (sal*0.1);
		}
		Employee e = new Employee(name, sal);
		emp.add(e);
	}
	System.out.println("Total bonus paid by the company is "+total);
	}

}

class Employee{
	String ename;
	int esal;
	public Employee(String ename, int esal) {
		this.ename = ename;
		this.esal = esal;
	}
}
```

Qu 3. Write a program using array to accept 10 numbers and display the numbers in ascending order.

```
package javaExercise;
import java.util.*;
public class Qu3 {

	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		int i, temp;
		int[] nos = new int[10];
		System.out.println("Enter 10 nos: ");
		for(i=0; i<10; i++) {
			nos[i] = s.nextInt();
		}
		for(i=0; i<10; i++) {
			for(int j=0; j<(9-i); j++) {
				if(nos[j]>nos[j+1]) {
					temp= nos[j];
					nos[j] = nos[j+1];
					nos[j+1] = temp;
				}
			}
		}
		System.out.println("Sorted Array: ");
		for(i=0; i<10; i++) {
			System.out.print(nos[i]+" ");
		}
	}

}
```

Qu 4. Write a program to accept 10 number using array and display the sum and average of 10 numbers.

```
package javaExercise;
import java.util.*;
public class Qu4 {

	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		int i, sum=0;
		float avg;
		int[] nos = new int[10];
		System.out.println("Enter 10 nos: ");
		for(i=0; i<10; i++) {
			nos[i] = s.nextInt();
			sum = sum + nos[i];
		}
		avg = sum/10;
		System.out.println("Sum: "+sum);
		System.out.println("Average: "+avg);
	}
}
```

Qu 5. Write a program to accept marks of 10 students using array and display the name of highest scorer.

```
package javaExercise;
import java.util.*;
public class Qu5 {

	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		int i,max=0;
		String topper="";
		String[] names = new String[10];
		int[] marks = new int[10];
		for(i=0; i<10; i++) {
			System.out.println("Enter name: ");
			names[i]=s.next();
			System.out.println("Enter marks: ");
			marks[i] = s.nextInt();
			if(marks[i] > max) {
				max = marks[i];
				topper = names[i];
			}
		}
		System.out.println(topper+" has scored the highest marks.");
	}

}
```

Qu 6. Write a java program to accept a number of any digit and reverse the numbers..

Like :- 678345 --> 543876

```
package javaExercise;
import java.util.*;
public class Qu6 {

	public static void main(String args[]) {
		Scanner s = new Scanner(System.in);
		int no,i;
		String rev="";
		System.out.println("Enter the no: ");
		no = s.nextInt();
		String in = Integer.toString(no);
		for(i=in.length()-1; i>=0; i--) {
			rev += in.charAt(i);
		}
		no = Integer.parseInt(rev);
		System.out.println("Reverse: "+no);
	}
}
```

Qu 7. Write a menu driven program for calculator using Java Language. Menu Details are mentioned below :-

1. Addition
2. Substraction
3. Multiplication
4. Division
5. Percentage
6. Exit

```
package javaExercise;
import java.util.*;
public class Qu7 {
	static Scanner s = new Scanner(System.in);

	static void add() {
		System.out.println("Enter two nos: ");
		int a= s.nextInt();
		int b = s.nextInt();
		System.out.println(a+" + "+b+" = "+(a+b));
	}
	static void sub() {
		System.out.println("Enter two nos: ");
		int a= s.nextInt();
		int b = s.nextInt();
		System.out.println(a+" - "+b+" = "+(a*b));
	}
	static void mul() {
		System.out.println("Enter two nos: ");
		int a= s.nextInt();
		int b = s.nextInt();
		System.out.println(a+" * "+b+" = "+(a*b));
	}
	static void div() {
		System.out.println("Enter two nos: ");
		int a= s.nextInt();
		int b = s.nextInt();
		System.out.println(a+" / "+b+" = "+(a/b));
	}
	static void per() {
		System.out.println("Enter two nos: ");
		float a= s.nextInt();
		float b = s.nextInt();
		float c = ((a/b)*100);
		System.out.println(a+" % "+b+" = "+c);
	}
```

```
	public static void main(String[] args) {
		int c=6;
		do{
			System.out.println("Select an option: \n1. Addition\n2.
			Subtraction\n3. Multiplication\n4. Division\n5. Percentage\n6.
			Exit");
			if(s.hasNextInt()) {
				c = s.nextInt();
			}
		else{
			System.out.print("Invalid choice. Select again.");
		}
		if(c==1) {
			add();
		}
		else if(c==2) {
			sub();
		}
		else if(c==3) {
			mul();
		}
		else if(c==4) {
			div();
		}
		else if(c==5) {
			per();
		}
		else{
			System.exit(0);
		}


		}while(true);

	}

}
```

Qu 8. Write a Java Program to accept the Name and Salary of five employees using array variable and perform below task.

1. Display the name of employee who is getting paid highest
2. Display the name of employee who is getting paid Lowest
3. Display the Average Salary of Employees.

```
package javaExercise;
import java.util.*;
public class Qu8 {

	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		int sum=0, maxsal=0, minsal=0;
		float avg;
		String hname="", lname="";
		int[] sal = new int[10];
		String[] name = new String[5];
		for(int i=0; i<5; i++) {
			System.out.println("Enter name: ");
			name[i]=s.next();
			System.out.println("Enter salary: ");
			sal[i] = s.nextInt();
			sum += sal[i];
			if(sal[i] > maxsal) {
				maxsal = sal[i];
				hname = name[i];
			}
			if(i==0) {
				minsal = sal[i];
				lname = name[i];
			}
			else {
				if (minsal > sal[i]) {
					minsal = sal[i];
					lname = name[i];
				}
			}
		}
		System.out.println(hname+" earns the most.");
		System.out.println(lname+" earns the least.");
		avg = sum / 5;
		System.out.println("Average salary of the employees is "+avg);


	}

}
```

Qu 9. Write a Java Program to display matrix of 3 rows and 3 columns.

```
package javaExercise;
//Matrix Display
public class QU9 {

	public static void main(String[] args) {
		int i,j;
		int[][] a = new int[3][3];
		for(i=0; i<3; i++) {
			for(j=0; j<3; j++) {
				a[i][j]=(int)(Math.floor(Math.random()*10));
			}
		}
		for(i=0; i<3; i++) {
			System.out.print("[ ");
			for(j=0; j<3; j++) {
				System.out.print(a[i][j]);
				if(j!=2)
					System.out.print(", ");
			}
			System.out.print(" ]");
			System.out.println("");
		}
	}

}
```

Qu 10. Write a Java Program to add 2 matrices of 3 rows and 3 columns.

```
package javaExercise;

public class Qu10 {

	public static void main(String[] args) {
		int i,j;
		int[][] a = new int[3][3];
		int[][] b = new int[3][3];
		int[][] c = new int[3][3];

		for(i=0; i<3; i++) {
			for(j=0; j<3; j++) {
				a[i][j]=(int)(Math.floor(Math.random()*10));
				b[i][j]=(int)(Math.floor(Math.random()*10));

			}
		}
		for(i=0; i<3; i++) {
			for(j=0; j<3; j++) {
				c[i][j]= a[i][j] + b[i][j];

			}
		}
		System.out.println("A-->");
		for(i=0; i<3; i++) {
			System.out.print("[ ");
			for(j=0; j<3; j++) {
				System.out.print(a[i][j]);
				if(j!=2)
					System.out.print(", ");
			}
			System.out.print(" ]");
			System.out.println("");
		}

		System.out.println("B-->");
		for(i=0; i<3; i++) {
			System.out.print("[ ");
			for(j=0; j<3; j++) {
				System.out.print(b[i][j]);
				if(j!=2)
					System.out.print(", ");
			}
			System.out.print(" ]");
			System.out.println("");
		}
		System.out.println("A+B=C-->");
		for(i=0; i<3; i++) {
			System.out.print("[ ");
			for(j=0; j<3; j++) {
```

```
                        System.out.print(c[i][j]);
                        if(j!=2)
                              System.out.print(", ");
                  }
                  System.out.print(" ]");
                  System.out.println("");
            }
      }

}
```

Qu 11. Write a Java Program to do product of 2 matrices of 3 rows and 3 columns.

```
package javaExercise;

public class Qu11 {

	public static void main(String[] args) {
		int i,j,k;
		int[][] a = new int[3][3];
		int[][] b = new int[3][3];
		int[][] c = new int[3][3];

		for(i=0; i<3; i++) {
			for(j=0; j<3; j++) {
				a[i][j]=(int)(Math.floor(Math.random()*10));
				b[i][j]=(int)(Math.floor(Math.random()*10));

			}
		}
		for (i = 0; i < 3; i++)
	 {
	     for (j = 0; j < 3; j++)
	     {
	         for (k = 0; k < 3; k++)
	         {
	             c[i][j] = c[i][j] + a[i][k] * b[k][j];
	         }
	     }
	 }
		System.out.println("A-->");
		for(i=0; i<3; i++) {
			System.out.print("[ ");
			for(j=0; j<3; j++) {
				System.out.print(a[i][j]);
				if(j!=2)
					System.out.print(", ");
			}
			System.out.print(" ]");
			System.out.println("");
		}

		System.out.println("\nB-->");
		for(i=0; i<3; i++) {
			System.out.print("[ ");
			for(j=0; j<3; j++) {
				System.out.print(b[i][j]);
				if(j!=2)
					System.out.print(", ");
			}
			System.out.print(" ]");
			System.out.println("");
		}
```

```
            System.out.println("\nA . B=C-->");
            for(i=0; i<3; i++) {
                  System.out.print("[ ");
                  for(j=0; j<3; j++) {
                        System.out.print(c[i][j]);
                        if(j!=2)
                              System.out.print(",  ");
                  }
                  System.out.print(" ]");
                  System.out.println("");
            }

      }

}
```

Qu 12. Write a java program to maintain Employee Details using ArrayList?

Id, Name, Age, Salary

1. Add minimum 5 employee details.

2. Display it in proper order.

3. Display the name to employee having highest Salary

4. Display the details in the order of Salary

```
package javaExercise;
import java.util.*;
public class Qu12 {

	public static void main(String[] args) {
		ArrayList<EmployeeDet> emp = new ArrayList<EmployeeDet>();

		EmployeeDet e1 = new EmployeeDet(101, "Tom", 27, 10000);
		EmployeeDet e2 = new EmployeeDet(103, "Jack", 32, 70000);
		EmployeeDet e3 = new EmployeeDet(105, "Rick", 42, 60000);
		EmployeeDet e4 = new EmployeeDet(102, "Ross", 24, 5000);
		EmployeeDet e5 = new EmployeeDet(104, "Sam", 36, 120000);

		emp.add(e1);
		emp.add(e2);
		emp.add(e3);
		emp.add(e4);
		emp.add(e5);

		//Sorting by employee id
		Collections.sort(emp);

		System.out.println("EmployeeDet Details Sorted by EmployeeDet ID.");
		for(EmployeeDet e: emp) {
			System.out.println("_____________________________________");
			System.out.println("ID: "+e.eid);
			System.out.println("Name: "+e.ename);
			System.out.println("Ager: "+e.age);
			System.out.println("Salary: "+e.sal);
			System.out.println("_____________________________________");
		}

		//sorting by salary
		Collections.sort(emp, EmployeeDet.sortSal);

		//displaying the employee details at the last index (max salary)
		System.out.println("EmployeeDet With The Highest Salary.");
		System.out.println("_____________________________________");
		System.out.println("ID: "+emp.get(emp.size()-1).eid);
```

```
			System.out.println("Name: "+emp.get(emp.size()-1).ename);
			System.out.println("Ager: "+emp.get(emp.size()-1).age);
			System.out.println("Salary: "+emp.get(emp.size()-1).sal);
			System.out.println("_____________________________________");


			System.out.println("EmployeeDet Details Sorted by Salary.");
			for(EmployeeDet e: emp) {
				System.out.println("_____________________________________");
				System.out.println("ID: "+e.eid);
				System.out.println("Name: "+e.ename);
				System.out.println("Ager: "+e.age);
				System.out.println("Salary: "+e.sal);
				System.out.println("_____________________________________");
			}
		}

}

class EmployeeDet implements Comparable<EmployeeDet>{
		int eid, age, sal;
		String ename;
		public EmployeeDet(int eid, String ename, int age, int sal){
			this.eid = eid;
			this.ename = ename;
			this.age = age;
			this.sal = sal;
		}

		public int compareTo(EmployeeDet e) {
			int id = e.eid;
			return this.eid - id;
		}

		public static Comparator<EmployeeDet> sortSal = new Comparator<EmployeeDet>()
{

			public int compare(EmployeeDet e1, EmployeeDet e2) {
				if(e1.sal>e2.sal) {
					return 1;
				}
				else if(e1.sal<e2.sal) {
					return -1;
				}
				else {
					return 0;
				}
			}
		};

}
```

Qu 13. Write a program in java to eliminate duplicate key in hash map as user defined object?

```
package javaExercise;
import java.util.*;
public class Qu13 {
	static HashMap<Integer, Emp> map = new HashMap<Integer, Emp>();

	static boolean keyExists(int eid) {
		//to check whether an entry with the given key already exists
		for(Map.Entry id: map.entrySet()){
			if(eid == (int) id.getKey()) {
				return true;
			}
		}
		return false;
	}

	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);

		do {
			System.out.println("Enter eid: ");
			int eid = s.nextInt();
			if(keyExists(eid)) {
				System.out.println("An entry with the given key already
exists please try again.");
				continue;
			}
			System.out.println("Enter name: ");
			String name = s.next();
			System.out.println("Enter salary: ");
			int sal = s.nextInt();


			Emp e = new Emp(eid, name, sal);
			map.put(eid, e);
			System.out.println("Entry added successfully.");
			System.out.println("HashMap after adding current entry: ");

			for(Map.Entry id: map.entrySet()) {
				eid = (int)id.getKey();
				e = (Emp)id.getValue();
				System.out.println(id.getKey() + "/t"+ e.ename +
"/t"+e.sal);
			}
			System.out.println("Continue(y/n):");
			String c = s.next();
			if(c.charAt(0)=='n' || c.charAt(0)=='N') {
				break;
			}
			else {
				continue;
			}
```

```
            }while(true);


        }

}

class Emp{
        String ename;
        int id, sal;
        public Emp(int id, String ename, int sal) {
                this.id = id;
                this.ename = ename;
                this.sal = sal;
        }

}
```

Qu 14. Write a program in Java to create Generic Class to accept Employee Age and Salary?

```
package javaExercise;

public class Qu14 {
	public static void main(String[] x) {
		EmpData<Integer, Integer> e = new EmpData<Integer, Integer>(22, 30000);

		e.printData();
	}

	static class EmpData<a, s>{
		private a age;
		private s sal;

		public EmpData(a age, s sal){
			this.age = age;
			this.sal = sal;
		}

		private void printData() {
			System.out.println("Age: "+age);
			System.out.println("Salary: "+sal);

		}

	}
}
```

Qu 15.

Write a program in Java to perform all the operations related with HashMap.

1. Store Employee Details by using Employee Class
2. Traverse Employee Details stored in Collection Object
3. Delete Employee Details
4. Update Employee Details

```
package javaExercise;
import java.util.*;
public class Qu15 {
	static HashMap<Integer, EmployeeData> empDet = new HashMap<Integer,
EmployeeData>();
	static Scanner s = new Scanner(System.in);
	static void traverseData() {
		if(empDet.size()==0) {
			System.out.println("Map is currently empty.");
		}

		else {
			for(Map.Entry<Integer, EmployeeData> data:empDet.entrySet()) {
				int key = data.getKey();
				EmployeeData e= data.getValue();

				System.out.println("_____________________________");
				System.out.println("Employee ID: "+key);
				System.out.println("Employee Name: "+e.ename);
				System.out.println("Age: "+e.age);
				System.out.println("Department: "+e.dept);
				System.out.println("Salary: "+e.sal);
				System.out.println("_____________________________");
			}
		}
	}
	static void enterData() {
		System.out.println("Enter Employee ID: ");
		int id = s.nextInt();
		if(empDet.containsKey(id)) {
			System.out.println("An entry with the given employee id already
exists. Please try again.");

		}
		else {
			System.out.println("Enter Employee Name: ");
			String name = s.next();
			System.out.println("Enter Employee Age: ");
			int age = s.nextInt();
```

```
                System.out.println("Enter Department Name: ");
                String dept = s.next();
                System.out.println("Enter Salary: ");
                int sal = s.nextInt();

                EmployeeData e = new EmployeeData(id, name, age, dept, sal);
                empDet.put(id, e);
            }


        }
        static void deleteData() {
            if(empDet.size()==0) {
                System.out.println("Map is currently empty.");

            }
            else {
                do {
                    System.out.println("Enter Employee ID Whose Data Is To Be
Deleted: ");
                    int id = s.nextInt();
                    if(empDet.containsKey(id)) {
                        empDet.remove(id);
                        System.out.println("Entry Deleted Successfully");
                        break;
                    }
                    else {
                        System.out.print("Entry With The Given ID Doesn't
Exists. ");

                        break;
                    }
                }while(true);
            }

        }

        static void updateData() {
            if(empDet.size()==0) {
                System.out.println("Map is currently empty.");

            }
            else {
                do {
                    System.out.println("Enter Employee ID Whose Data Is To Be
Updated: ");
                    int id = s.nextInt();
                    if(!empDet.containsKey(id)) {
                        System.out.print("Entry With The Given ID Doesn't
Exists.");
                    }
                    else{

                        empDet.remove(id);
                        System.out.println("Enter Employee Name: ");
                        String name = s.next();
```

```
                                System.out.println("Enter Employee Age: ");
                                int age = s.nextInt();
                                System.out.println("Enter Department Name: ");
                                String dept = s.next();
                                System.out.println("Enter Salary: ");
                                int sal = s.nextInt();

                                EmployeeData e = new EmployeeData(id, name, age,
dept, sal);
                                empDet.put(id, e);
                                System.out.println("Update Successfull");
                                break;
                        }

                }while(true);
        }

    }
    public static void main(String[] args) {
        int c=0;
        do {
                System.out.println("Select an option: \n1. Enter New Data\n2.
View Employee Data.\n3. Delete Data.\n4. Update Employee details.\n5. Exit");
                if(s.hasNextInt()) {

                        c = s.nextInt();
                        if(c==1) {
                                enterData();
                        }
                        else if(c==2) {
                                traverseData();
                        }
                        else if(c==3) {
                                deleteData();
                        }
                        else if(c==4) {
                                updateData();
                        }
                        else if(c==5){
                                System.exit(0);
                        }
                }

        }while(c>=0 && c<5);
    }

}

class EmployeeData {
    String ename, dept;
    int id, age, sal;
    public EmployeeData(int id, String ename, int age, String dept, int sal) {
        this.id = id;
        this.ename = ename;
        this.age = age;
```

```
                this.dept = dept;
                this.sal = sal;
        }
}
```

Qu 16.
Write a Java Program to Accept Employee Details by Using TreeSet. Enter the data in any order but display the data by arranging as per EmployeeID using Comparator interface.
1. Employee Name
2. Employee ID
3. Employee Age

```
import java.util.Collections;
import java.util.Comparator;
import java.util.TreeSet;

public class Qu16 {

	public static void main(String[] args) {
		TreeSet<Employee1> emp = new TreeSet<Employee1>(new sortEmployee1());
		Employee1 e1 = new Employee1(102, "Jimmy", 28);
		Employee1 e2 = new Employee1(104, "Abel", 23);
		Employee1 e3 = new Employee1(105, "Cecil", 32);
		Employee1 e4 = new Employee1(103, "Nadia", 26);
		Employee1 e5 = new Employee1(101, "Mary", 42);

		emp.add(e1);
		emp.add(e2);
		emp.add(e3);
		emp.add(e4);
		emp.add(e5);



		for(Employee1 e: emp) {
			System.out.println("_____________________________");
			System.out.println("ID: "+e.id);
			System.out.println("Name: "+e.name);
			System.out.println("Age: "+e.age);
			System.out.println("_____________________________");
		}

	}

}

class Employee1{
	String name;
	int id, age;
	public Employee1(int id, String name, int age) {
		this.id = id;
		this.name = name;
		this.age = age;
	}
}
```

```
class sortEmployee1 implements Comparator<Employee1>{
	public int compare(Employee1 e1, Employee1 e2) {
		if(e1.id > e2.id) {
			return 1;
		}
		else if(e1.id < e2.id) {
			return -1;
		}
		else {
			return 0;
		}
	}
}
```

Qu 17.

Write program in java to create a base class name for accepting Studnet Details of Science Stream with common properties. After that create child classes of various Science Stream like (Maths, Biology, Computer, Electronics)

After that also create the another child class (Software,Hardware),(Botany,Zoology) on class like Computer and Biology.

```
package javaExercise;
public class Qu17 {

	public static void main(String[] args) {
		Zoology z = new Zoology();
		z.printDetails();
		Maths m = new Maths();
		Elec e = new Elec();
		z.bio();
		Botany b = new Botany();
		b.bio();
		Hardware hw = new Hardware();
		Software sw = new Software();
		hw.com();
		sw.com();


	}

}

class StudentDetails{
	public  void printDetails() {
		System.out.println("Name: Tom");
		System.out.println("ID: 101");
		System.out.println("Contact No: 9999999999");
	}


}

class Maths extends StudentDetails{
	public Maths() {
		System.out.println("Marks for maths: 85");
	}
}
class Elec extends StudentDetails{
	public Elec() {
		System.out.println("Marks for electronics: 82");
	}
}

abstract class Biology extends StudentDetails{
	public abstract void bio();
```

```
}

class Botany extends Biology{
	public void bio() {
		System.out.println("Marks for botany: 78");
	}
}

class Zoology extends Biology{
	public void bio() {
		System.out.println("Marks for zoology: 87");
	}
}

abstract class Computer extends StudentDetails{
	public abstract void com();
}

class Hardware extends Computer{
	public void com() {
		System.out.println("Marks for hardware: 79");
	}
}

class Software extends Computer{
	public void com() {
		System.out.println("Marks for software: 86");
	}
}
```