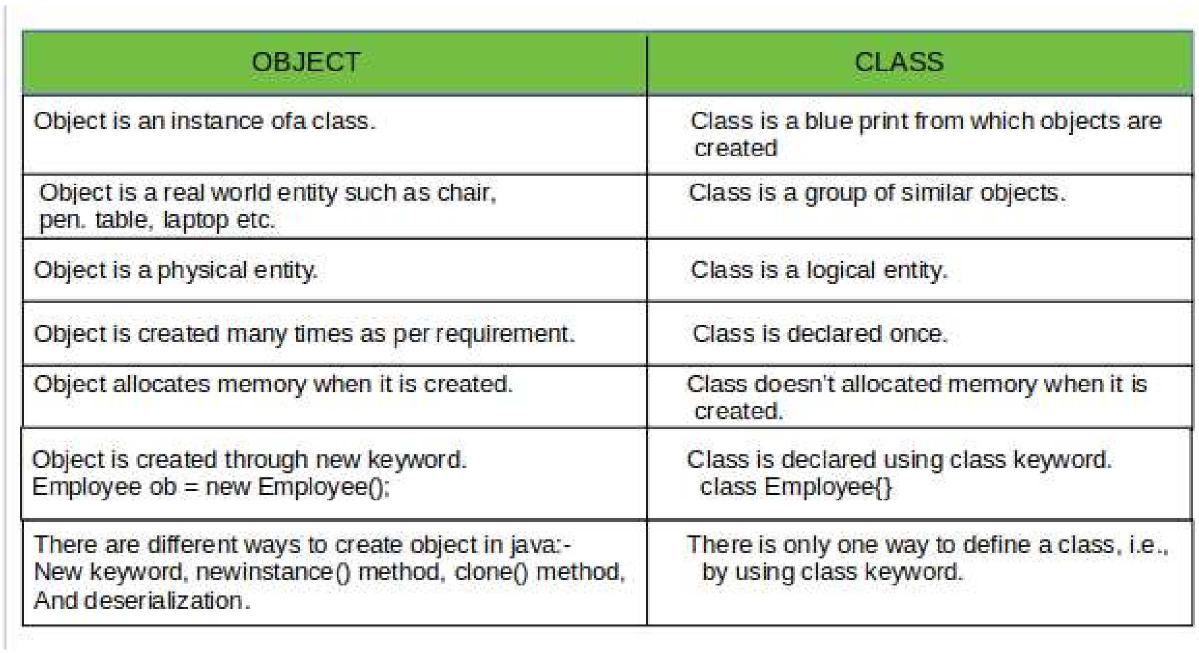

OBJECT	CLASS
Object is an instance ofa class.	Class is a blue print from which objects are created
Object is a real world entity such as chair, pen. table, laptop etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.
Object is created through new keyword. Employee ob = new Employee();	Class is declared using class keyword. class Employee{}
There are different ways to create object in java:- New keyword, newinstance() method, clone() method, And deserialization.	There is only one way to define a class, i.e., by using class keyword.

Question : Create an abstract class named Account for a bank. Include an integer field for the account number and a double field for the account balance. Include instance variables for account holder's details. Also include a constructor that requires an account number and a balance value and sets these values to the appropriate fields. Add two public get methods for the fields – getAccountNumber() and getAccountBalance() – each of which returns the appropriate field. Also add an abstract method named display().

Create two subclasses of Account: Current and Saving. Within the Current class, the display method displays the string "Current Account Information", the account number and the balance. Within the Saving class, add a field to hold the interest rate and require the Saving constructor to accept an argument for the value of the interest rate. The Saving display method displays the string "Saving Account Information", the account number, the balance and the interest rate.

Example

Read in an integer command-line argument n. Then, read in a list of n student records from standard input into a Student data type. Each record consists of four fields, separated by whitespace:

- *first name*
- *last name*
- *email address*
- *which section they're in*

Then, print out the list of the first N students in sorted order, according to their section number.

```
public class Student {
    private String first;     // first name
    private String last;      // last name
    private String email;     // email address
    private int section;      // section number

    // construct a new student with given fields
    public Student(String first, String last, String email, int section) {
        this.first   = first;
        this.last    = last;
        this.email   = email;
        this.section = section;
    }

    // return true if the invoking object's section is less than that of b
    public boolean less(Student b) {
        Student a = this;
        return a.section < b.section;
    }

    // return a string representation of the invoking object
    public String toString() {
        return section + " " + first + " " + last + " " + email;
    }



    // sample client
    public static void main(String[] args) {
        // number of students
        int n = Integer.parseInt(args[0]);

        // initialize an arary that holds n objects of type Student
        Student[] students = new Student[n];

        // read in the data
        for (int i = 0; i < n; i++) {
            String first   = StdIn.readString();
            String last    = StdIn.readString();
            String email   = StdIn.readString();
            int section    = StdIn.readInt();
            students[i] = new Student(first, last, email, section);
        }

        // insertion sort students in ascending order of section
        for (int i = 0; i < n; i++) {
            for (int j = i; j > 0; j--) {
                if (students[j].less(students[j-1])) {
                    Student swap  = students[j];
                    students[j]   = students[j-1];
                    students[j-1] = swap;
                }
            }
        }

        // print results
        for (int i = 0; i < n; i++) {
            StdOut.println(students[i]);
        }
    }

}
```

Question
Consider the following definition of the class MyClass:

```
public class MyClass
{
  private static int count = 0;
  private int x;

  public MyClass(int i)
  {
    x = i;
  }
  public void incrementCount()
  {
    count++;
  }

  public void printX()
  {
    System.out.println("Value of x : "  + x);
  }

  public static void printCount()
  {
    System.out.println("Value of count : "  + count);
  }
}

public class MyClassDemo
{
  public static void main(String[] args)
  {
      MyClass myObject1 = new MyClass(5);
      MyClass myObject2 = new MyClass(7);
  }
}
```

What is the output of the following Java code? (Assume that following statements are written inside main)

```
myObject1.printX();
myObject1.incrementCount();
```

```
MyClass.incrementCount();
myObject1.printCount();
myObject2.printCount();
myObject2.printX();
myObject1.setX(14);
myObject1.incrementCount();
myObject1.printX();
myObject1.printCount();
myObject2.printCount();
```

Question

Write a Java class Clock for dealing with the day time represented by hours, minutes, and seconds. Your class must have the following features:

- Three instance variables for the hours (range 0 - 23), minutes (range 0 - 59), and seconds (range 0 - 59).
- Three constructors:
    - default (with no parameters passed; is should initialize the represented time to 12:0:0)
    - a constructor with three parameters: hours, minutes, and seconds.
    - a constructor with one parameter: the value of time in seconds since midnight (it should be converted into the time value in hours, minutes, and seconds)
- Instance methods:
    - a *set*-method method setClock() with one parameter *seconds* since midnight (to be converted into the time value in hours, minutes, and seconds as above).
    - *get*-methods getHours(), getMinutes(), getSeconds() with no parameters that return the corresponding values.
    - *set*-methods setHours(), setMinutes(), setSeconds() with one parameter each that set up the corresponding instance variables.
    - method tick() with no parameters that increments the time stored in a Clock object by one second.
    - method addClock() accepting an object of type Clock as a parameter. The method should add the time represented by the parameter class to the time represented in the current class.
    - Add an instance method toString() with no parameters to your class. toString() must return a String representation of the Clock object in the form "(hh:mm:ss)", for example "(03:02:34)".
    - Add an instance method tickDown() which decrements the time stored in a Clock object by one second.
    - Add an instance method subtractClock() that takes one Clock parameter and returns the difference between the time represented in the current Clock object and the one represented by the Clock parameter. Difference of time should be returned as an clock object.

Write a separate class ClockDemo with a main() method. The program should:

- instantiate a Clock object firstClock using one integer *seconds* since midnight obtained from the keyboard.
- tick the clock ten times by applying its *tick()* method and print out the time after each tick.
- Extend your code by appending to it instructions instantiating a Clock object secondClock by using three integers (hours, minutes, seconds) read from the keyboard.
- Then tick the clock ten times, printing the time after each tick.

- Add the secondClock time in firstClock by calling method addClock.
- Print both clock objects calling toString method

Create a reference thirdClock that should reference to object of difference of firstClock and secondClock by calling the method subtractClock().

Question

Write a Java class Complex for dealing with complex number. Your class must have the following features:

- Instance variables :
  - **realPart** for the real part of type double
  - **imaginaryPart** for imaginary part of type double.
- Constructor:
  - **public Complex ()**: A default constructor, it should initialize the number to 0, 0)
  - **public Complex (double realPart, double imaginaryPart)**: A constructor with parameters, it creates the complex object by setting the two fields to the passed values.
- Instance methods:
  - **public Complex add (Complex otherNumber)**: This method will find the sum of the current complex number and the passed complex number. The methods returns a new Complex number which is the sum of the two.
  - **public Complex subtract (Complex otherNumber)**: This method will find the difference of the current complex number and the passed complex number. The methods returns a new Complex number which is the difference of the two.
  - **public Complex multiply (Complex otherNumber)**: This method will find the product of the current complex number and the passed complex number. The methods returns a new Complex number which is the product of the two.
  - **public Complex divide (Complex otherNumber)**: This method will find the ... of the current complex number and the passed complex number. The methods returns a new Complex number which is the ... of the two.
  - **public void setRealPart (double realPart)**: Used to set the real part of this complex number.
  - **public void setImaginaryPart (double realPart)**: Used to set the imaginary part of this complex number.
  - **public double getRealPart()**: This method returns the real part of the complex number
  - **public double getImaginaryPart()**: This method returns the imaginary part of the complex number
  - **public String toString()**: This method allows the complex number to be easily printed out to the screen

Write a separate class **ComplexDemo** with a main() method and test the Complex class methods.

Question

Write a Java class Author with following features:

- Instance variables :
  - **firstName** for the author’s first name of type String.
  - **lastName** for the author’s last name of type String.
- Constructor:
  - **public Author (String firstName, String lastName)**: A constructor with parameters, it creates the Author object by setting the two fields to the passed values.
- Instance methods:
  - **public void setFirstName (String firstName)**: Used to set the first name of author.
  - **public void setLastName (String lastName)**: Used to set the last name of author.
  - **public double getFirstName()**: This method returns the first name of the author.
  - **public double getLastName()**: This method returns the last name of the author.
  - **public String toString()**: This method printed out author’s name to the screen

Write a Java class Book with following features:

- Instance variables :
  - **title** for the title of book of type String.
  - **author** for the author’s name of type String.
  - **price** for the book price of type double.
- Constructor:
  - **public Book (String title, Author name, double price)**: A constructor with parameters, it creates the Author object by setting the the fields to the passed values.
- Instance methods:
  - **public void setTitle(String title)**: Used to set the title of book.
  - **public void setAuthor(String author)**: Used to set the name of author of book.
  - **public void setPrice(double price)**: Used to set the price of book.
  - **public double getTitle()**: This method returns the title of book.
  - **public double getAuthor()**: This method returns the author’s name of book.
  - **public String toString()**: This method printed out book’s details to the screen

Write a separate class **BookDemo** with a main() method creates a Book titled “Developing Java Software” with authors Russel Winderand price 79.75. Prints the Book’s string representation to standard output (using System.out.println).