01.
a. Class/ Template: A class is a blueprint for creating objects. It defines the properties and methods that all objects of that type will have. For example, the Box class defines the properties length, width, and height, and the method getVolume().

b. Object/ Instance: An object is an instance of a class. It has its own unique values for its properties, and it can be manipulated by calling its methods. For example, the box1 object is an instance of the Box class. It has its own length, width, and height, and it can be used to calculate its volume by calling the getVolume() method.

c. Methods/ Functions: A method is a block of code that performs a specific task. It is associated with a class, and it can be called on objects of that type. For example, the getVolume() method is a method of the Box class. It can be called on any Box object to calculate its volume.

d. Attributes/ Properties: An attribute or property is a characteristic of an object. It can be a data value, such as the length of a Box, or it can be a method, such as the getVolume() method of a Box.

e. Reference Variables: A reference variable is a variable that stores a reference to an object. It is used to access the object's properties and methods. For example, the box1 variable is a reference variable. It stores a reference to the Box object.

f. Primitive Variables: A primitive variable is a variable that stores a primitive data type, such as an integer, a floating-point number, or a character. For example, the length variable is a primitive variable. It stores an integer value.

g. Method Parameters: A method parameter is a variable that is used to pass data into a method. The method can then use this data to perform its task. For example, the getVolume() method has a method parameter called length. This parameter is used to pass the length of a Box object into the method.

h. Local Variables: A local variable is a variable that is declared within a method. It is only accessible within the method in which it is declared. For example, the volume variable is a local variable. It is declared within the getVolume() method, and it is only accessible within that method.

i. Default Values: A default value is a value that is assigned to a variable if no other value is specified. For example, the default value for a primitive variable is 0. The default value for a reference variable is null.

j. Declaration Values: A declaration value is the value that is assigned to a variable when it is declared. For example, the declaration value for the length variable is 10.

k. Constructor: A constructor is a special method that is used to create an object. It is called when an object is created, and it is used to initialize the object's properties. For example, the Box class has a constructor that takes three parameters: length, width, and height. These parameters are used to initialize the length, width, and height properties of the Box object.

l. Instance Blocks: An instance block is a block of code that is executed when an object is created. It is used to initialize the object's properties. For example, the Box class has an instance block that initializes the length, width, and height properties of the Box object to 0.

m. Static Variables: A static variable is a variable that is associated with a class, rather than with an object. It is shared by all objects of that class. For example, the pi variable is a static variable. It is associated with the Math class, and it is shared by all objects of the Math class.

n. Instance Variables: An instance variable is a variable that is associated with an object. It is not shared by other objects of the same class. For example, the length variable is an instance variable. It is associated with the Box class, but it is not shared by other Box objects.

02. D

03.89

04.
Attributes are data that is stored in an object. They are like the properties of an object. For example, a Person object might have attributes like name, age, and address.
Methods are actions that an object can perform. They are like the functions of an object. For example, a Person object might have methods like getAge(), setAge(), and sayHello().
Here are some real-world examples of attributes and methods:

Attributes:
A car might have attributes like color, make, and model.
A book might have attributes like title, author, and genre.
A person might have attributes like name, age, and address.
Methods:
A car might have methods like start(), stop(), and turn().
A book might have methods like open(), close(), and read().
A person might have methods like speak(), walk(), and eat().

05.iligal

06.all out put is 0.

07.
```
Volume : 180
length of box : 12
```

```
width of box : 5
height of box : 3
```

08.constructor eken valus Initialize karagannawa

09.
```
Parameterized constructor
Volume : 60
Parameterized constructor
Volume : 180
```

10.width attribute eka private nisa e class eken eliye access karanna be

11. error

12.
A default constructor is a constructor that does not have any parameters. It is automatically created by the Java compiler if no other constructors are defined in the class.

A parameterized constructor is a constructor that has one or more parameters. It is used to initialize the object's state with the values of the parameters.

13.
Constructors are used to initialize an object when it is created. They are similar to methods, but they do not have a return type. For example, a constructor for a Car object might initialize the car's make, model, and year.

Methods are used to perform actions on an object. They can have a return type, or they can be void. For example, a method for a Car object might start the car, stop the car, or turn on the radio.

14.B

15.line 2 , 3 error

16.C

17.A

18.
```
100 101
0 0
```

19.Code : 3001

20.Code : 2001

21.A

22.D

23.

24.

25.D

26.
```
//--------------------Date.java------------------------
class Date{
	private int year=1970;
	private int month=1;
	private int day=1;

	public void printDate(){
		System.out.println(year+" - "+month+" - "+day);
		}

	public void setYear(int year){
		this.year=year;
		}

	public void setMonth(int month){
		this.month=month;
		}

	public void setDay(int day){
		this.day=day;
		}

	public int getYear(){
		return this.year;
		}
	public int getMonth(){
		return this.month;
		}
	public int getDay(){
		return this.day;
```

```
        }
    }
//--------------------Demo.java------------------------
class Example{
    public static void main(String args[]){
            Date d1=new Date();
            d1.printDate(); //1970-1-1
            //d1.year=2016;//Illegal
            //d1.month=5; //Illegal
            //d1.day=30; //Illegal
/*year, month and day attributes cannot be accessed to another class*/
            d1.setYear(2016);
            d1.setMonth(5);
            d1.setDay(31);
            System.out.println("Year : "+d1.getYear());
            System.out.println("Month :"+d1.getMonth());
            System.out.println("Day : "+d1.getDay());
    }
}
```

27.

28.compile error at all lines

29.E

30.
31.A,C

32.1 200 10 200 100 200

33.line 5 and line 8

34.
Static variables are declared using the static keyword. They are shared by all instances of the class and are not associated with any specific object. Static variables are initialized when the class is loaded and destroyed when the class is unloaded.

Instance variables are declared without the static keyword. They are associated with each individual object and are not shared by other objects of the same class. Instance variables are initialized when an object is created and destroyed when the object is destroyed.

35.A D F I

36.

```
public class MyDate extends Date {

    public void set(int field, int value) {
        switch (field) {
            case Date.YEAR:
                this.year = value;
                break;
            case Date.MONTH:
                this.month = value;
                break;
            case Date.DAY:
                this.day = value;
                break;
        }
    }

    public void printDate() {
        System.out.println(this.year + "-" + this.month + "-" + this.day);
    }
}
```

37.

Static methods are declared using the static keyword. They are associated with the class itself, not with any specific object. Static methods can be called without creating an instance of the class.

Instance methods are declared without the static keyword. They are associated with each individual object and can only be called on an object. Instance methods must have an implicit this parameter, which refers to the current object.

38.

out put is :

```
Box is loaded into memory
```

This is because the static block is executed when the Box class is loaded into memory, even before any instances of the class are created. The static block can be used to initialize static variables or perform other tasks that need to be done when the class is loaded.

In this case, the static block simply prints a message to the console to indicate that the Box class has been loaded.

39.
```
A box object is created..
A box object is created..
```

The reason for this is because the Box class has a constructor that prints the message "A box object is created.." when it is called. When the main() method creates two objects of type Box, the constructor is called twice, once for each object.

40.
```
class Cylinder {

    private double radius;
    private double height;

    public Cylinder() {
        this.radius = 0.0;
        this.height = 0.0;
    }

    public Cylinder(double radius, double height) {
        this.radius = radius;
        this.height = height;
    }

    public double getRadius() {
        return radius;
    }

    public void setRadius(double radius) {
        this.radius = radius;
    }

    public double getHeight() {
        return height;
    }

    public void setHeight(double height) {
        this.height = height;
    }
    public double getVolume() {
        return Math.PI * radius * radius * height;
    }

    public double getArea() {
        return 2 * Math.PI * radius * (radius + height);
    }

    @Override
    public String toString() {
        return "Cylinder [radius=" + radius + ", height=" + height + "]";
    }
}
public class Example {

    public static void main(String[] args) {
        Cylinder cylinder1 = new Cylinder();
        System.out.println("Cylinder1: " + cylinder1);

        Cylinder cylinder2 = new Cylinder(5, 10);
        System.out.println("Cylinder2: " + cylinder2);

        System.out.println("The volume of cylinder1 is " + cylinder1.getVolume());
        System.out.println("The area of cylinder1 is " + cylinder1.getArea());

        System.out.println("The volume of cylinder2 is " + cylinder2.getVolume());
        System.out.println("The area of cylinder2 is " + cylinder2.getArea());
    }
}
```

41.

```
Box is loaded into memory
A box object is created..
A box object is created..
A box object is created..
```

The reason for this output is that the static block is executed only once, when the class is loaded into memory. The instance block is executed every time a new object of the class is created. In this case, three new objects of the Box class are created, so the instance block is executed three times.

Here is a more detailed explanation of what happens when the code is executed:

The class Box is loaded into memory.
The static block is executed. This prints the message "Box is loaded into memory".
Three new objects of the Box class are created.
For each new object, the instance block is executed. This prints the message "A box object is created..".
The main method ends.

42.
C
D

43.

A. new MyClass();
Output: 1 2 3

B. new MyClass(); new MyClass();
Output: 1 2 3 2 3

C. MyClass.mB();
Output: 1 2 4

D. new MyClass().mA();
Output: 1 2 3 Error: MyClass does not have a method called mA()

E. MyClass.mB();
Output: 1 2 4

44.
(5, 3)(3, 5)

45.
In method call by value, a copy of the argument is passed to the method. Any changes made to the argument inside the method do not affect the original argument.

46.
Line 3: The cat variable is declared as a Cat object, but it is initialized with an array of Cat objects. This is illegal because an array cannot be assigned to a variable of a single object type.

Line 5: The cats variable is declared as an array of Cat objects, but it is initialized with a single Cat object. This is illegal because an array cannot be assigned to a variable of a single object type.

47.

48.

```
class Employee {

    private String firstName;
    private String lastName;
    private double monthlySalary;

    public Employee(String firstName, String lastName, double monthlySalary) {
        this.firstName = firstName;
        this.lastName = lastName;
        setMonthlySalary(monthlySalary);
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public double getMonthlySalary() {
        return monthlySalary;
    }

    public void setMonthlySalary(double monthlySalary) {
        if (monthlySalary > 0) {
            this.monthlySalary = monthlySalary;
        }
    }

    public double getYearlySalary() {
        return monthlySalary * 12;
    }
    public void giveRaise(double percentage) {
        monthlySalary += monthlySalary * percentage / 100;
    }

    @Override
    public String toString() {
        return String.format("Employee: %s %s, monthly salary: %.2f", firstName, lastName,
monthlySalary);
    }
}


class Example{

    public static void main(String[] args) {
        Employee employee1 = new Employee("John", "Doe", 10000);
        Employee employee2 = new Employee("Jane", "Smith", 20000);

        System.out.println(employee1);
        System.out.println(employee2);

        employee1.giveRaise(10);
        employee2.giveRaise(15);

        System.out.println("After giving raises:");
        System.out.println(employee1);
        System.out.println(employee2);
    }
}
```

49.

```
code : 1001
code : 1001
```

50.

```
class ExtendedDate extends Date {

    public ExtendedDate() {
        super();
    }

    public ExtendedDate(int year, int month, int day) {
        super(year, month, day);
    }
    public void set(int field, int value) {
        switch (field) {
            case Date.YEAR:
                this.year = value;
                break;
            case Date.MONTH:
                this.month = value;
                break;
            case Date.DAY:
                this.day = value;
                break;
            default:
                throw new IllegalArgumentException("Invalid field: " + field);
        }
    }

    public void printDate() {
        System.out.println(this.year + "-" + this.month + "-" + this.day);
    }
}
```

51.B

52.

```
MyClass()
MyClass(int)
MyClass(int,int)
```

53.

```
class Box {

    private int length;
    private int width;
    private int height;

    public Box() {
        this.length = 0;
        this.width = 0;
        this.height = 0;
    }

    public Box(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    public int getLength() {
        return this.length;
    }

    public void setLength(int length) {
        this.length = length;
    }

    public int getWidth() {
        return this.width;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public int getHeight() {
        return this.height;
    }

    public void setHeight(int height) {
        this.height = height;
    }

    public int getVolume() {
        return this.length * this.width * this.height;
    }

    public void printVolume() {
        System.out.println("Volume: " + getVolume());
    }

    public void setDimension(int length, int width, int height) {
        this.length = length;
        this.width = width;
        this.height = height;
    }

    public Box getCopy() {
        return new Box(this.length, this.width, this.height);
    }
}
```