Date :

(s) (s) (m) (t) (w) (t) (f)

1. Class and object

```
class Car {
        string color;
        int   speed;

        void drive(){
             system. out. println("car  is driving");
        }
  }

 public class main{
             public static void main(string[] args)

 {     car myCar = new Car();
      myCar. color="Red";
      mycar . speed = 100 ;
      mycar .drive();
 }  }
```

Date

2. Access Modifier :

```
class person {
      private string name;

      public void setName(string newName) {
            name = newName;
      }

      public string getName() {
            return name;
      }
}
public class main {
      public class void main(string[] args){
            person p = new person();
            p. setName = ~~new~~ "Ramizkhan";
      sout(p. setName);
      }
}
```

Date:
ⓢ ⓢ ⓜ ⓣ ⓦ ⓣ ⓕ

3. Inheritance & protected ~~on~~ Access

```
class Animal {
        protected string type = "Animal";
        void display() {
            system. out. println("This is an
                                        Animal");
   } }
class Dog extends Animal {
        void bark() {
            system. out. println(type+ "says
                                        woof");
  } }
public class Main {
        public static void main(String[] args)
 {      Dog d = new Dog();
        d.display();
        d. bark();
  }
}
```

4. Encapsulation:

```
class BankAccount {
        private double balance;
```

Date :

```
public void deposit (double amount){
        if(amount >0) balance += amount;
    }

  public double getBalance(){
        return balance;
    }
}

public class main {
    public static void main(String[] args){

BankAccount acc = new BankAccount();
    acc.deposite(500);
    System.out.println(acc.getbalance());
  }
}
```

5. Abstract class

```
Abstract class Animal {
     Abstruct void makesound();
    void sleep(){
        System.out.println("~~Bark Bark~~ sleeping..");
} }
```

Date
Ⓢ Ⓢ Ⓜ Ⓣ Ⓦ Ⓣ Ⓕ

```
class Dog extends Animal {
    void makesound(){
        system.out. println("Bark Bark");
} }

public class main {
   public static void main(String[] args)
                                          {
       Dog d = new Dog();
        d. ~~makesound~~ makesound ();
         d. sleep();
}  }
```

6. Interface

```
interface Animal {
        void sound();
   }
class    cat implements Animal {
     public void sound() {
       System.out. println("Meow");
 }  }
```

Date:
Ⓢ Ⓢ Ⓜ Ⓣ Ⓦ Ⓣ Ⓕ

```
public class main {
    public static void main(String[] args){

        cat c = new cat();

        c.sound();
    }
}
```

7. Multiple Inheritance using Interface

```
interface flyable {
        void fly();
}
interface swimmable {
        void swim();
    }
class Duck implements flyable, swimmable {
    public void fly () {
        System.out.println("Duck is flying");
}
public void swim{
    System. out. println(" Duck is swimmin
                                    ");
  }
}

public class main{
    public static void main(String[] args)
  {
      Duck d = new Duck();
        d. fly ();
        d .swim();
      }
}
```

8. ATM (mini projects

```
import java.util. scanner;

public class ATM {
    private double balance=5000.0
    public void deposite(double amount)
    {
      balance += amount;
    }
```

Date
Ⓢ Ⓢ Ⓜ Ⓣ Ⓦ Ⓣ Ⓕ

```
public void witAdaw (double amount){
    if(amount <= balance){
        balance -= amount;
    }else {
        System.out.println("Insufficiet
                            balance");
    }
}

public void checkbalance(){
    System.out.println("cunnent balance"+
                        balance);
}
public static void main(String[] args){
    ATM atm= new ATM();
    Scanner sc = new Scanner(System.in);
    while (true) {
        System.out.println ("\n1. Deposite
                            2. withdaw
                            3. balance
                            4. exit");
```

Date :

ⓢ ⓢ ⓜ ⓣ ⓦ ⓣ ⓕ

```
int choice ~~s~~ = sc. NextInt();

switch (choice) {

   case 1:
        System.out.print("Enter amount:");
        atm. deposite (sc. nextDouble());
        break;

   case 2:
        System. out. print("Enter amount:");
        atm ~~.~~ withdaw (sc. nextDouble());
        break;

  case 3:
        atm. checkBalance ();
        break;

case 4:
        System. ~~out~~ exit(0);
   }
  }
 }
}
```

Date
ⓢ ⓢ ⓜ ⓣ ⓦ ⓣ ⓕ

9. Calculator (mini project)

```
import java. util. Scanner;
public class calculator {
  public static void main (String[] args) {
  Scanner sc = new Scanner(system.in);
  System.out. print("Enter first number:");
   double Num1 = Sc. NextDouble();
 System. out. print("Enter Second Number:");
  double Num2 = Sc. NextDouble();
System. out.println("Choose operation
                  +, -, *, /):");
  Char op= sc. Next(). CharAt(0);
  double result = 0;
switch (op) {
      case '+': result = Num1 + Num2; break;
```

Date:

```
case '-' : result = Num1 - Num2; break;

case '*' : result = Num1 * Num2; break;

case '/' :
        if (num2 != 0)
            result = ~~new~~ Num1 / Num2;

      else {
        System.out.println("cannot divide
                            by zero");
      }
      break;

default:
      System.out.println(" Invalid operation
                                        ");
 }

 System.out.println("Result: " + result);

}

}
```