Practical 01.

01)
```
package com.mycompany.practical1;
public class Practical1
{
   public static void main(String[] args)
   {
      System.out.println("Hello World!");
   }
}
```

02)
```
package com.mycompany.practical1;
public class Practical1
{
   public static void main(String[] args)
   {
      System.out.println("Sapna Dilupshi");
      System.out.println("Bsc (Honours) Softare Engineering");
   }
}
```

03) <u>For Loop</u>
```
package com.mycompany.practical1;
public class Practical1
{
    public static void main(String[] args)
   {
      for(int i=0;i<5;i++)
      {
         System.out.println("Executing Loop "+i);
      }
   }
}
```

<u>While Loop</u>
```
package com.mycompany.practical1;
public class Practical1
{
   public static void main(String[] args)
   {
```

```
        int i=0;
        while(i<5)
        {
           System.out.println("Executing Loop "+i);
           i++;
        }
     }
}
```

04) Result 1.

10
20
I'm out of the Loop now

Result 2.

10
20
40
50
I'm out of the Loop now

05) Result 1.

Excellent!
Your grade is A

Result 2.

Excellent!
You passed
Better try again
Your grade is A

Using if-else-if Statement.

```
package com.mycompany.practical1;
public class Practical1
{

   public static void main(String[] args)
   {
      char grade='A';

      if(grade=='A')
         System.out.println("Excellent!");
      else if (grade=='D')
         System.out.println("You passed");
      else if (grade=='F')
         System.out.println("Better try again");
      else
         System.out.println("Invalid grade");

      System.out.println("Your grade is "+grade);
   }
}
```

06) Code.

```
package com.mycompany.testenhanceforloop;
public class TestEnhanceForLoop
{

   public static void main(String args[])
   {
      int[] numbers={10,20,30,40,50};
      for(int x: numbers)
      {
         System.out.print(x);
         System.out.print(",");
      }
      System.out.print("\n");
      String [] names={"James","Larry","Tom","Lacy"};
            for(String name: names)
            {
               System.out.print(name);
               System.out.print(",");
            }
   }
}
```

Output.
10,20,30,40,50,
James,Larry,Tom,Lacy,

Practical 02.

PART 01

```
package com.mycompany.itemobj;
public class Item
{
   //Data
   private int location;
   private String description;
   //Parameterized constructor
   public Item (int location,String description)
   {
      this.location=location;
      this.description=description;
   }
   //setter method to location
   public void setLocation(int lo)
   {
      location=lo;
   }
   //getter method to location
   public int getLocation()
   {
      return location;
   }
   //setter method to Description
```

```
    public void setDescription(String de)
    {
        description=de;
    }
    //getter method to description
    public String getDescription()
    {
        return description;
    }
    public void display()
    {
        System.out.println("The location is: "+location);
        System.out.println("The description is: "+description);
    }
}
```

……..

```
package com.mycompany.itemobj;
public class Monster extends Item
{
    public Monster(int location,String description)
    {
        super (location,description);
    }
}
```

……

```
package com.mycompany.itemobj;

public class ItemObj
{
```

```
    public static void main(String[] args)
    {
        Item i1 =new Item(14,"Colombo");
        System.out.println("Item location is: "+i1.getLocation());
        System.out.println("Item description is: "+i1.getDescription());

        Monster m1 = new Monster(74,"Kandy");
        System.out.println("Monster location is: "+m1.getLocation());
        System.out.println("Monster description is: "+m1.getDescription());
    }
}
```

Output.

Item location is: 14

Item description is: Colombo

Monster location is: 74

Monster description is: Kandy

PART 02

1. Which of these keywords is used to refer to member of base class from a sub class?
   a) upper **b) super** c) this d) None of the mentioned

2. The modifier which specifies that the member can only be accessed in its own class is
   a) public **b) private** c) protected d) none

3. Which of these is a mechanism for naming and visibility control of a class and its content?
   a) Object b) Packages
   c) Interfaces **d) None of the Mentioned.**

4. Which of the following is correct way of importing an entire package ‘pkg’?
   a) import pkg. b) Import pkg.
   c) import pkg.* d) Import pkg.*

5. Which of these method of class String is used to extract a single character from a String object?
   a) CHARAT() b) charat()
   c) charAt() d) CharAt()

6. Which of these method of class String is used to obtain length of String object?
   a) get() b) Sizeof()
   c) lengthof() **d) length()**

PART 03: Fill in the blanks using appropriate term.

1. Real-world objects contain **state(Data)** and **behavior(Method).**
2. software object's state is stored in **instance variables(Fields).**
3. A software object's behavior is exposed through **methods.**
4. Hiding internal data from the outside world, and accessing it only through publicly exposed methods is known as **data encapsulation.**
5. A blueprint for a software object is called **a class.**
6. Common behavior can be defined in a **super class** and inherited into a **sub class** using the **extends** keyword.
7. A collection of methods with no implementation is called an **interface**.
8. A namespace that organizes classes and interfaces by functionality is called a **package.**
9. The term API stands for? **Application Programming Interface.**

Practical 02.

Practical 03.

Exercise 3.1

01. Setter method and getter method

```
package com.mycompany.employeeobj;
public class Employee
{
   //Data declaration
   private String name;
   private int age;
   private double salary;

   //setter method to Employee Name
   public void setName(String name)
   {
      this.name=name;
   }
   //getter mmethod to Employee Name
   public String getName()
   {
      return name;
   }
   //setter method to age
   public void setAge(int age)
   {
      this.age=age;
```

```
    }
    //getter method to age
    public int getAge()
    {
        return age;
    }
    //setter method to salary
    public void setSalary(double salary)
    {
        this.salary=salary;
    }
    //getter method to salary
    public double getSalary()
    {
        return salary;
    }
}
```

……..

```
package com.mycompany.employeeobj;
public class EmployeeObj
{

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

```
        Employee e1 = new Employee();

        e1.setName("Anne Teesha");

        e1.setAge(21);

        e1.setSalary(30000);

        System.out.println("Employee Name is: "+e1.getName());

        System.out.println("Employee Age is: "+e1.getAge());

        System.out.println("Employee Salary is: Rs."+e1.getSalary());

    }

}
```

Output.

```
Employee Name is: Anne Teesha

Employee age is: 21

Employee Salary is: Rs.30000.0
```

02. Constructor method

```
    package com.mycompany.employeeobj;
    public class Employee
    {
      //Data declaration
      private String name;
      private int age;
      private double salary;

      //parameterized constructor
      public Employee(String name,int age,double salary)
      {
        this.name=name;
        this.age=age;
        this.salary=salary;
      }
      public String getName()
      {
```

```
        return name;
    }
    public int getAge()
    {
        return age;
    }
    public double getSalary()
    {
        return salary;
    }
}
```

……

```
package com.mycompany.employeeobj;
public class EmployeeObj
{

    public static void main(String[] args)
    {
        Employee e1 = new Employee("Anne Teesha",21,30000);

        System.out.println("Employee Name is: "+e1.getName());
        System.out.println("Employee age is: "+e1.getAge());
        System.out.println("Employee Salary is: Rs."+e1.getSalary());
    }
}
```

Output.

```
Employee Name is: Anne Teesha
Employee age is: 21
Employee Salary is: Rs.30000.0
```

Exercise 3.2

```
package com.mycompany.employeeobj1;
public class Employee
{
   //Employee data
   private String name;
   private double basicSalary;
   private double bonus;

   //Constructor
   public Employee(String name, double basicSalary, double bonus)
   {
      this.name = name;
      this.basicSalary = basicSalary;
      this.bonus = bonus;
   }
   //Getter method for name
   public String getName()
   {
      return name;
   }
   //Setter method for name
   public void setName(String name)
   {
      this.name = name;
   }
   //Getter method for Employee Salary
   public double getBasicSalary()
   {
      return basicSalary;
```

```
    }
    //Setter method for Employee Salary
    public void setBasicSalary(double basicSalary)
    {
        this.basicSalary = basicSalary;
    }
    //Getter method for Bonus
    public double getBonus()
    {
        return bonus;
    }
    //Display Employee Details
    public void displayEmployeeInfo()
    {
        System.out.println("Employee Name: " + getName());
        System.out.println("Basic Salary: " + getBasicSalary());
        System.out.println("Bonus: " + getBonus());
    }
}
```

……

```
package com.mycompany.employeeobj1;
public class EmployeeObj1
{
    public static void main(String[] args)
    {
        Employee employee = new Employee("Anne Teesha", 60000.0, 5000.0);
        employee.displayEmployeeInfo();
    }
}
```

Output.

Employee Name: Anne Teesha

Basic Salary: 60000.0

Bonus: 5000.0

Exercise 3.3

```
package com.mycompany.employeeobj3;
public class Employee
{
   //Employee Data
   private String name;
   private double basicSalary;
   private double bonus;

   //Parameterized Constructor
   public Employee(String name, double basicSalary, double bonus)
   {
      this.name = name;
      this.basicSalary = basicSalary;
      this.bonus = bonus;
   }
   //Getter method for Employee name
   public String getName()
   {
      return name;
   }
   //Setter method for Employee name
   public void setName(String name)
   {
```

```
    this.name = name;
}
//Getter method for Basic Salary
public double getBasicSalary()
{
    return basicSalary;
}
//Setter method for Basic Salary
public void setBasicSalary(double basicSalary)
{
    this.basicSalary = basicSalary;
}
//Getter method for Bonus
public double getBonus()
{
    return bonus;
}
//Setter method for Bonus
public void setBonus(double bonus)
{
    this.bonus = bonus;
}
//Calulate
public double calculateBonusAmount()
{
    return getBasicSalary() + getBonus();
}
//Display Employee Details
public void displayEmployeeInfo()
```

```
    {
        System.out.println("Employee Name: " + getName());
        System.out.println("Basic Salary: " + getBasicSalary());
        System.out.println("Bonus: " + getBonus());
        System.out.println("Bonus Amount: " + calculateBonusAmount());
    }
}
```

……

```
package com.mycompany.employeeobj3;
public class EmployeeObj3
{
    public static void main(String[] args)
    {
        Employee employee = new Employee("Bogdan", 50000.0, 10000.0);
        employee.displayEmployeeInfo();
    }
}
```

Practical 04.

01.

```
package com.mycompany.employeeobj2;
public class Employee
{
    private int empID;
    private String empName;
    private String empDesignation;
```

```
//setter method to Employee Id
public void setEmpID(int empID)
{
   this.empID=empID;
}
//setter method to Employee Name
public void setEmpName(String empName)
{
   this.empName=empName;
}
//setter method to Employee Designation
public void setEmpDesignation(String empDesignation)
{
   this.empDesignation=empDesignation;
}
//getter method to Employee ID
public int getEmpID()
{
   return empID;
}
//getter method to Employee Name
public String getEmpName()
{
   return empName;
}
public String getEmpDesignation()
{
   return empDesignation;
}
```

```
}
```

……

```
package com.mycompany.employeeobj2;
public class EmployeeObj2
{

   public static void main(String[] args)
   {
      Employee MrBogdan=new Employee();
      MrBogdan.setEmpID(1012);
      MrBogdan.setEmpName("Mr.Bogdan");
      MrBogdan.setEmpDesignation("Manager");

      Employee MsBird=new Employee();
      MsBird.setEmpID(1014);
      MsBird.setEmpName("Ms.Bird");
      MsBird.setEmpDesignation("Engineer");

      System.out.println("Employee ID: "+MrBogdan.getEmpID());
      System.out.println("Employee Name: "+MrBogdan.getEmpName());
      System.out.println("Employee Designation: "+MrBogdan.getEmpDesignation());

      System.out.println("Employee ID: "+MsBird.getEmpID());
      System.out.println("Employee Name: "+MsBird.getEmpName());
      System.out.println("Employee Designation: "+MsBird.getEmpDesignation());
   }
}
```

Output.

```
Employee ID: 1012
```

Employee Name: Mr.Bogdan

Employee Designation: Manager

Employee ID: 1014

Employee Name: Ms.Bird

Employee Designation: Engineer

02.

Output.

9

6

03.

04.

Output.

True

True

True

Practical 05.

Exercise 01.

01) There is no difference between these approaches. Because the interface automatically applies public, static, final keywords to the variable declaration.

02) There is no difference between these approaches. When the methods declared within an interface are implicitly include the “abstract” keyword. All methods within an interface are considered abstract keyword by default.

03) No, x is not possible to change. Because override method is not support in interface. When we try to change “x” value then result will give in a compilation error.

```
package com.mycompany.practical6;
public interface MyFirstInterface
{
   //data
   int x=5;
   void display();
}
```

```
package com.mycompany.practical6;
public class InterfaceImplemented implements MyFirstInterface
{
   @Override
   public void display()
   {
      //x=10;
      System.out.println("x: "+x);
   }
}
```

```
package com.mycompany.practical6;
public class Practical6
{

   public static void main(String[] args)
   {
      InterfaceImplemented inface = new InterfaceImplemented();
      inface.display();
```

```
        }
    }
```

Exercise 02.

```
package com.mycompany.practical5ii;
public interface Speaker
{
   void speak();
}
```

```
package com.mycompany.practical5ii;
public class Politician implements Speaker
{
  @Override
  public void speak()
  {

  }
}
```

```
package com.mycompany.practical5ii;
public class Priest implements Speaker
{
```

```
    @Override
    public void speak()
    {

    }
}
```

```
package com.mycompany.practical5ii;
public class Lecture
{
    public void speak()
    {

    }
}
```

```
package com.mycompany.practical5ii;
public class Practical5ii
{

    public static void main(String[] args)
    {
        Priest sp1 = new Priest();
        Politician sp2 = new Politician();
        Lecture sp3 = new Lecture();

        sp1.speak();
        sp2.speak();
        sp3.speak();
```

```
    }
}
```