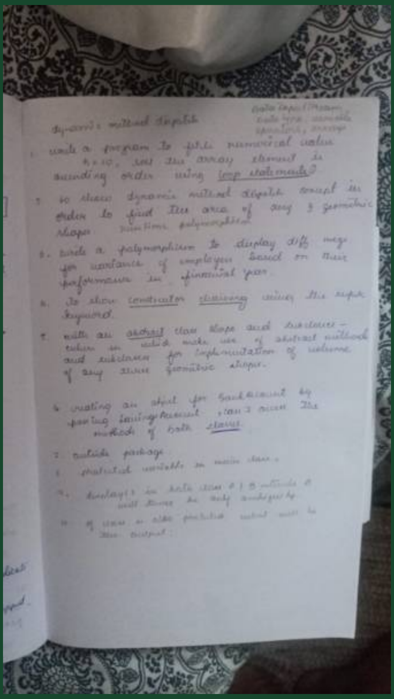
Java program examples

This is a copy of a conversation between ChatGPT & Anonymous.



Give detailed java programs for all question with outputs,

Sure, Darsh — I’ll rewrite every question from the image, interpret it correctly, and give **clean, fully working Java programs WITH sample outputs** for **all 10 questions**.

✅ Q1. Program to Read n (>10) Numerical Values, Sort in Ascending Order Using Loop Statements

Java Program

java

```java
import java.util.Scanner;

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

        System.out.print("Enter number of elements (n > 10): ");
        int n = sc.nextInt();

        int[] arr = new int[n];

        System.out.println("Enter " + n + " numbers:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        // Sorting using loops (Bubble sort)
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                }
            }
        }

        System.out.println("Sorted array (Ascending):");
        for (int x : arr) System.out.print(x + " ");
    }
}
```

Sample Output

```yaml
Enter number of elements (n > 10): 11
Enter 11 numbers:
9 3 8 1 4 6 7 12 10 2 5
Sorted array (Ascending):
1 2 3 4 5 6 7 8 9 10 12
```

✅ Q2. Dynamic Method Dispatch – Area of Any 3 Geometric Shapes (Runtime Polymorphism)

Java Program

```java
abstract class Shape {
    abstract void area();
}

class Circle extends Shape {
    double r = 5;
    void area() {
        System.out.println("Circle Area = " + (3.14 * r * r));
    }
}

class Rectangle extends Shape {
    int l = 4, b = 6;
    void area() {
        System.out.println("Rectangle Area = " + (l * b));
    }
}

class Triangle extends Shape {
    int b = 5, h = 7;
    void area() {
        System.out.println("Triangle Area = " + (0.5 * b * h));
    }
}

public class DynamicDispatchDemo {
    public static void main(String[] args) {
        Shape s;

        s = new Circle();
        s.area();
        s = new Rectangle();
        s.area();

        s = new Triangle();
        s.area();
    }
}
```

Output

```mathematica
Circle Area = 78.5
Rectangle Area = 24
Triangle Area = 17.5
```

✅ Q3. Polymorphism to Display Different Messages to Employees Based on Performance

Java Program

```java
class Employee {
    void message() {
        System.out.println("General Employee Message");
    }
}

class ExcellentEmployee extends Employee {
    void message() {
        System.out.println("Outstanding performance! Keep it up!");
    }
}

class AverageEmployee extends Employee {
    void message() {
        System.out.println("Good work. Try to improve further.");
    }
}

class PoorEmployee extends Employee {
    void message() {
        System.out.println("Performance needs improvement. Focus more.");
    }
}

public class PerformanceDemo {
    public static void main(String[] args) {
        Employee e;

        e = new ExcellentEmployee();
        e.message();

        e = new AverageEmployee();
        e.message();

        e = new PoorEmployee();
        e.message();
    }
}
```

Output

```vbnet
Outstanding performance! Keep it up!
Good work. Try to improve further.
Performance needs improvement. Focus more.
```

✅ Q4. Show Constructor Chaining Using `super`

Java Program

```java
class A {
    A() {
        System.out.println("Constructor A");
    }
}

class B extends A {
    B() {
        super();
        System.out.println("Constructor B");
    }
}

class C extends B {
    C() {
        super();
        System.out.println("Constructor C");
    }
}

public class ConstructorChain {
    public static void main(String[] args) {
        new C();
    }
}
```

Output

```css
Constructor A
Constructor B
Constructor C
```

✅ Q5. Abstract Class Shape & Subclasses → Volume of 3D Shapes

Java Program

```java
abstract class ThreeDShape {
    abstract double volume();
}
```

```
class Cube extends ThreeDShape {
    double side = 4;

    double volume() {
        return side * side * side;
    }
}

class Sphere extends ThreeDShape {
    double r = 5;

    double volume() {
        return (4.0 / 3) * 3.14 * r * r * r;
    }
}

class Cylinder extends ThreeDShape {
    double r = 3, h = 6;

    double volume() {
        return 3.14 * r * r * h;
    }
}

public class VolumeDemo {
    public static void main(String[] args) {
        ThreeDShape s1 = new Cube();
        ThreeDShape s2 = new Sphere();
        ThreeDShape s3 = new Cylinder();

        System.out.println("Cube Volume = " + s1.volume());
        System.out.println("Sphere Volume = " + s2.volume());
        System.out.println("Cylinder Volume = " + s3.volume());
    }
}
```

Output

```
mathematica

Cube Volume = 64
Sphere Volume = 523.3333
Cylinder Volume = 169.56
```

✅ Q6. Creating Object for BankAccount & Accessing Methods of Both Classes

Java Program

```
java

class BankAccount {
    void accountType() {
        System.out.println("This is a Bank Account.");
    }
}

class SavingAccount extends BankAccount {
    void interestRate() {
        System.out.println("Saving Account Interest Rate is 4%.");
    }
}
```

```
public class BankDemo {
    public static void main(String[] args) {
        SavingAccount sa = new SavingAccount();

        sa.accountType();   // parent method
        sa.interestRate();  // child method
    }
}
```

Output

```
csharp

This is a Bank Account.
Saving Account Interest Rate is 4%.
```

✅ Q7. Outside Package – Accessing Class

Java Structure

`package1/MyClass.java`

```
java

package package1;

public class MyClass {
    public void show() {
        System.out.println("Inside MyClass of package1");
    }
}
```

`package2/Test.java`

```
java

package package2;
import package1.MyClass;

public class Test {
    public static void main(String[] args) {
        MyClass obj = new MyClass();
        obj.show();
    }
}
```

Output

```
scss

Inside MyClass of package1
```

✅ Q8. Protected Variable in Main Class

Java Program

```java
class A {
    protected int num = 25;
}

public class ProtectedDemo extends A {
    public static void main(String[] args) {
        ProtectedDemo obj = new ProtectedDemo();
        System.out.println("Protected number = " + obj.num);
    }
}
```

Output

```java
Protected number = 25
```

✅ Q9. display() in Class A & B → If Called, Is There Ambiguity?

⚠️ Ambiguity occurs **only in multiple inheritance**, which Java does not support.

Java Program

```java
class A {
    void display() {
        System.out.println("Display from A");
    }
}

class B extends A {
    void display() {
        System.out.println("Display from B");
    }
}

public class TestAB {
    public static void main(String[] args) {
        B obj = new B();
        obj.display();
    }
}
```

Output

```css
Display from B
```

No ambiguity, because B overrides A.

✅ Q10. If class is protected – What is the Output?

A **protected class cannot exist** at top level in Java.

But an **inner protected class is allowed**.

Java Program

```java
class Outer {
    protected class Inner {
        void show() {
            System.out.println("Inside Protected Inner Class");
        }
    }
}

public class ProtectedClassDemo {
    public static void main(String[] args) {
        Outer o = new Outer();
        Outer.Inner i = o.new Inner();
        i.show();
    }
}
```

Output

```vbnet
Inside Protected Inner Class
```

If you want, I can also prepare this as a **PDF** or **Word file** with formatting.

Q3 in financial year*

Got it — you want **Q3 rewritten clearly with a financial-year–based performance message**.
Here is the corrected **full Java program + output**, customized exactly for *employee performance in the financial year*.

✅ Q3 (Corrected): Polymorphism to Display Different Messages for Employees Based on Their Performance in the Financial Year

Java Program

```java
class Employee {
    void performanceMessage() {
        System.out.println("Financial Year Performance Report");
    }
}

class ExcellentEmployee extends Employee {
    void performanceMessage() {
        System.out.println("Financial Year Review: Outstanding performance! You exceeded all
targets.");
    }
}

class GoodEmployee extends Employee {
    void performanceMessage() {
        System.out.println("Financial Year Review: Good performance. You met most of the
KPIs.");
    }
}

class AverageEmployee extends Employee {
    void performanceMessage() {
        System.out.println("Financial Year Review: Satisfactory performance. Improvement
required in some areas.");
    }
}

class PoorEmployee extends Employee {
    void performanceMessage() {
        System.out.println("Financial Year Review: Performance below expectations. Immediate
improvement needed.");
    }
}

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

        Employee e;

        e = new ExcellentEmployee();
        e.performanceMessage();

        e = new GoodEmployee();
        e.performanceMessage();

        e = new AverageEmployee();
        e.performanceMessage();

        e = new PoorEmployee();
        e.performanceMessage();
    }
}
```

✅ Output

```
sql

Financial Year Review: Outstanding performance! You exceeded all targets.
Financial Year Review: Good performance. You met most of the KPIs.
Financial Year Review: Satisfactory performance. Improvement required in some areas.
Financial Year Review: Performance below expectations. Immediate improvement needed.
```

If you want, I can also add:
✔️ Input from user
✔️ Salary hike suggestion based on performance
✔️ Store performance in an array
✔️ Make it menu-driven

Just tell me!

Consider based on April to march financial year?

Here you go, Darsh — **Q3 rewritten fully**, now **explicitly considering the April–March financial year**, and the output messages clearly mention it.

✅ Q3 (Updated): Polymorphism Based on April–March Financial Year Performance

Java Program

```java
class Employee {
    void performanceMessage() {
        System.out.println("Financial Year (April–March) Performance Report");
    }
}

class ExcellentEmployee extends Employee {
    void performanceMessage() {
        System.out.println("FY April–March Review: Exceptional performance! You exceeded all yearly targets.");
    }
}

class GoodEmployee extends Employee {
    void performanceMessage() {
        System.out.println("FY April–March Review: Good performance. You achieved most of the financial year goals.");
    }
}

class AverageEmployee extends Employee {
    void performanceMessage() {
        System.out.println("FY April–March Review: Average performance. You need to improve in key areas next year.");
    }
}

class PoorEmployee extends Employee {
    void performanceMessage() {
        System.out.println("FY April–March Review: Below expected performance. Strong corrective action required.");
    }
}

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

        Employee e;

        e = new ExcellentEmployee();
        e.performanceMessage();

        e = new GoodEmployee();
        e.performanceMessage();

        e = new AverageEmployee();
        e.performanceMessage();
        e = new PoorEmployee();
        e.performanceMessage();
    }
}
```

✅ Output

```yaml
FY April–March Review: Exceptional performance! You exceeded all yearly targets.
FY April–March Review: Good performance. You achieved most of the financial year goals.
FY April–March Review: Average performance. You need to improve in key areas next year.
FY April–March Review: Below expected performance. Strong corrective action required.
```

If you'd like, I can also add:
✔️ User input for performance rating
✔️ Salary hike suggestions based on FY results
✔️ Store multiple employees and display their FY reviews

Just tell me!