perplexity

Java Programming Concepts – Step-by-Step Answers

Basic Programs

1. Cube of a Number

Question: How can you compute the cube of a number in Java?

Answer:
To compute the cube of a number, multiply the number by itself twice:

```
int n = 3;
int cube = n * n * n;
System.out.println("Cube of " + n + " is " + cube);
```

Output: Cube of 3 is 27

2. Area and Perimeter of a Circle

Question: How do you calculate the area and perimeter (circumference) of a circle?

Answer:

```
double radius = 5.0;
double area = Math.PI * radius * radius;
double perimeter = 2 * Math.PI * radius;
System.out.println("Area: " + area);
System.out.println("Perimeter: " + perimeter);
```

Output: Area: 78.54, Perimeter: 31.42 (rounded values)

3. Swapping Two Numbers (with and without Third Variable)

With a Third Variable:

```
int a = 5, b = 10, temp;
temp = a;
a = b;
b = temp;
```

Without a Third Variable:

```
int a = 5, b = 10;
a = a + b; // a=15
b = a - b; // b=5
a = a - b; // a=10
```

4. Even or Odd Number

Question: How do you check if a number is even or odd?

Answer:

```
int n = 4;
if (n % 2 == 0)
    System.out.println("Even");
else
    System.out.println("Odd");
```

5. Factorial of a Number

Question: How is the factorial of a number calculated?

Answer:

```
int n = 5, fact = 1;
for(int i=1;i<=n;i++){
    fact *= i;
}
System.out.println("Factorial: " + fact);
```

6. Fibonacci Series

Question: How to print the Fibonacci series up to n terms?

Answer:

```
int n = 5, a=0, b=1;
System.out.print(a + " " + b);
for(int i=2;i<n;i++){
    int c = a+b;
    System.out.print(" " + c);
    a = b;
    b = c;
}
```

Output: 0 1 1 2 3

7. Prime Number Check

Question: How to check if a number is prime?

Answer:

```
int n = 7, count = 0;
for(int i=2;i<=n/2;i++){
    if(n%i==0){
        count++;
        break;
    }
}
if(count==0 && n>1)
    System.out.println("Prime");
else
    System.out.println("Not Prime");
```

8. Palindrome Number

Question: How do you check if a number is a palindrome?

Answer:

```
int n = 121, rev=0, temp=n;
while(temp>0){
    rev = rev*10 + temp%10;
    temp = temp/10;
}
if(n==rev)
    System.out.println("Palindrome");
else
    System.out.println("Not Palindrome");
```

9. Reverse a Number

Question: How do you reverse a number in Java?

Answer:

```
int n = 1234, rev=0;
while(n>0){
    rev = rev*10 + n%10;
    n = n/10;
}
System.out.println("Reversed: " + rev);
```

10. Sum of Digits in a Number

Question: How is the sum of the digits of a number calculated?

Answer:

```
int n = 123, sum=0;
while(n>0){
    sum += n%10;
    n = n/10;
}
System.out.println("Sum of digits: " + sum);
```

Arrays

1. Array Sorting (Ascending & Descending)

Ascending Order:

```
Arrays.sort(arr);
```

Descending Order:

```
Arrays.sort(arr, Collections.reverseOrder());
```

2. Search an Element in an Array

Answer:

```
int[] arr = {1, 3, 5, 7};
int key = 5, found = -1;
for(int i=0;i<arr.length;i++){
    if(arr[i]==key){
        found = i; break;
    }
}
if(found!=-1)
    System.out.println("Found at index " + found);
else
    System.out.println("Not found");
```

3. Find the Maximum or Minimum Element

Maximum:

```
int max = arr[0];
for(int i=1;i<arr.length;i++){
    if(arr[i]>max)
```

```
        max = arr[i];
}
```

Minimum:

```
int min = arr[0];
for(int i=1;i<arr.length;i++){
    if(arr[i]<min)
        min = arr[i];
}
```

4. Sum of All Array Elements

Answer:

```
int sum=0;
for(int v : arr)
    sum += v;
System.out.println("Sum: " + sum);
```

5. Merge Two Arrays

Answer:

```
int[] a = {1,2}, b = {3,4};
int[] merged = new int[a.length + b.length];
System.arraycopy(a,0,merged,0,a.length);
System.arraycopy(b,0,merged,a.length,b.length);
```

OOPs Concepts

1. Data Hiding Using Private Keyword

Question: What is data hiding and how is it implemented?

Answer:
Data hiding restricts access to class members by declaring them as `private`, so they're not directly accessible outside the class. Access is provided via public methods (getters/setters).

2. Encapsulation

Question: What does encapsulation mean in OOP?

Answer:
Encapsulation is the concept of binding data (variables) and methods that operate on the data into a single unit, i.e., a class. It hides the internal state of the object from outside.

3. Inheritance (Single and Multilevel)

Single Inheritance:
A derived class inherits from a single base class.
Multi-level Inheritance:
A class derives from another derived class, forming a chain.

```
class A {}
class B extends A {}
class C extends B {}
```

4. Polymorphism (Method Overloading/Overriding)

Overloading:
Same method name, different parameters in same class.

```
void display(int a);
void display(String s);
```

Overriding:
Subclass provides specific implementation for a method in parent class.

```
@Override
void display() { ... }
```

5. Abstraction (Interfaces and Abstract Classes)

- **Abstraction** hides the implementation details and shows only the essential features.
- **Abstract class**: Can have abstract and concrete methods.
- **Interface**: All methods are abstract (in Java 8, can have default methods).

```
abstract class Shape { abstract void draw(); }
interface Drawable { void draw(); }
```

main() Method

1. Signature of main()

Question: What is the signature of the main() method in Java?

Answer:
`public static void main(String[] args)`

2. What Happens if main() is Missing or Misdeclared

Answer:
If the main() method is missing or its signature is incorrect, the Java Virtual Machine won't recognize the entry point and throws a `NoSuchMethodError` or execution error.

3. Why main() is `public static void`

Answer:

- **public**: Accessible to JVM from anywhere.
- **static**: Does not require an object to run.
- **void**: Does not return any value.

□ JDBC Basics

1. What is JDBC?

Answer:
JDBC (Java Database Connectivity) is an API that enables Java applications to connect and execute queries with databases.

2. Why Persistence is Needed

Answer:
Persistence allows data to be stored and retrieved even after the application is closed or restarted, ensuring data longevity and consistency.

3. JDBC Drivers

Types:

- Type 1: JDBC-ODBC Bridge
- Type 2: Native-API
- Type 3: Network Protocol Driver
- Type 4: Thin Driver

4. Steps to Connect Java with DB

1. Load JDBC driver
2. Establish connection using `DriverManager`
3. Create `Statement` object
4. Execute SQL queries
5. Process results
6. Close connection

JavaPoint JDBC Tutorial
GeeksforGeeks JDBC Drivers