

# Object Oriented Programming with JAVA (4341602) - Winter 2024 Solution

Milav Dabgar

November 26, 2024

## પ્રશ્ન 1(અ) [3 ગુણ]

OOP અને POP વચ્ચેનો તફાવત લખો.

જવાબ

કોષ્ટક 1. OOP vs POP

પાસાં	OOP	POP
અભિગમ	બોટમ-અપ અભિગમ	ટોપ-ડાઉન અભિગમ
ફોકસ	ઓબ્જેક્ટ અને ક્લાસ	ફંક્શન અને પ્રોસીજર
ડેટા સિક્યોરિટી	એન્કેપ્સ્યુલેશન દ્વારા ડેટા હાઇડિંગ	ડેટા હાઇડિંગ નથી
પ્રોબ્લેમ સોલ્વિંગ	સમસ્યાને ઓબ્જેક્ટમાં વિભાજિત કરો	સમસ્યાને ફંક્શનમાં વિભાજિત કરો

મેમરી ટ્રીક

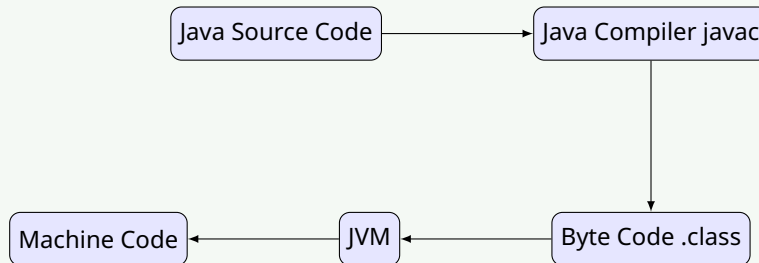
"Objects Bottom, Procedures Top"

## પ્રશ્ન 1(બ) [4 ગુણ]

બાઇટ કોડ શું છે? JVM ને વિગતવાર સમજાવો.

જવાબ

બાઇટ કોડ (Byte Code): Java compiler દ્વારા સોર્સ કોડમાંથી જનરેટ થતો પ્લેટફોર્મ-ઇન્ડિપેન્ડન્ટ ઇન્ટરપ્રીટ થતો કોડ.



JVM કોમ્પોનન્ટ્સ:

- **Class Loader:** .class ફાઇલોને મેમરીમાં લોડ કરે છે
- **Memory Area:** Heap, stack, method area સ્ટોરેજ
- **Execution Engine:** બાઇટકોડને ઇન્ટરપ્રેટ અને એક્ઝિક્યુટ કરે છે
- **Garbage Collector:** ઓટોમેટિક મેમરી મેનેજમેન્ટ

## મેમરી ટ્રીક

"Byte Code Runs Everywhere"

## પ્રશ્ન 1(ક) [7 ગુણ]

એરેના એલિમેન્ટ્સને ચડતા ક્રમમાં સોર્ટ કરવા માટે જાવામાં પ્રોગ્રામ લખો

## જવાબ

Listing 1. Array Sort

```

1 import java.util.Arrays;
2
3 public class ArraySort {
4     public static void main(String[] args) {
5         int[] arr = {64, 34, 25, 12, 22, 11, 90};
6
7         // Bubble Sort
8         for(int i = 0; i < arr.length-1; i++) {
9             for(int j = 0; j < arr.length-i-1; j++) {
10                 if(arr[j] > arr[j+1]) {
11                     int temp = arr[j];
12                     arr[j] = arr[j+1];
13                     arr[j+1] = temp;
14                 }
15             }
16         }
17
18         System.out.println("Sorted array: " + Arrays.toString(arr));
19     }
20 }

```

## મુખ્ય મુદ્દાઓ:

- Bubble Sort: બાજુના એલિમેન્ટ્સની તુલના કરે છે
- Time Complexity:  $O(n^2)$
- Space Complexity:  $O(1)$

## મેમરી ટ્રીક

"Bubble Up The Smallest"

## પ્રશ્ન 1(ક OR) [7 ગુણ]

કમાંડ લાઇન આર્ગ્યુમેન્ટ્સનો ઉપયોગ કરીને કોઈપણ દસ સંખ્યાઓમાંથી મહત્તમ શોધવા માટે જાવામાં પ્રોગ્રામ લખો.

## જવાબ

Listing 2. Find Maximum from Command Line

```

1 public class FindMaximum {
2     public static void main(String[] args) {
3         if(args.length != 10) {
4             System.out.println("Please enter exactly 10 numbers");
5             return;
6         }
7     }
8 }

```

```

6      }
7
8      int max = Integer.parseInt(args[0]);
9
10     for(int i = 1; i < args.length; i++) {
11         int num = Integer.parseInt(args[i]);
12         if(num > max) {
13             max = num;
14         }
15     }
16
17     System.out.println("Maximum number: " + max);
18 }
19 }

```

મુખ્ય મુદ્દાઓ:

- **Command Line:** args[] array આર્ગ્યુમેન્ટ્સ સ્ટોર કરે છે
- **parseInt():** સ્ટ્રિંગને ઇન્ટિજરમાં કન્વર્ટ કરે છે
- **Validation:** Array length ચેક કરો

મેમરી ટ્રીક

``Arguments Maximum Search``

## પ્રશ્ન 2(અ) [3 ગુણ]

Wrapper ક્લાસ શું છે? ઉદાહરણ સાથે સમજાવો.

જવાબ

**Wrapper Class:** પ્રિમિટિવ ડેટા ટાઇપ્સને ઓબ્જેક્ટમાં કન્વર્ટ કરે છે.

કોષ્ટક 2. Wrapper Classes

Primitive	Wrapper Class
int	Integer
char	Character
boolean	Boolean
double	Double

Listing 3. Wrapper Class Example

```

1 // Boxing
2 Integer obj = Integer.valueOf(10);
3 // Unboxing
4 int value = obj.intValue();

```

મેમરી ટ્રીક

``Wrap Primitives Into Objects``

## પ્રશ્ન 2(બ) [4 ગુણ]

જાવાના વિવિધ લક્ષણોની યાદી આપો. કોઈપણ બે સમજાવો.

### જવાબ

#### Java Features:

- **Simple:** સરળ syntax, pointers નથી
- **Platform Independent:** એકવાર લખો, દરેક જગ્યાએ ચલાવો
- **Object Oriented:** ઓબ્જેક્ટ અને ક્લાસ પર આધારિત
- **Secure:** explicit pointers નથી, bytecode verification

#### વિગતવાર સમજૂતી:

- **Platform Independence:** Java bytecode JVM વાળા કોઈપણ પ્લેટફોર્મ પર ચાલે છે
- **Object Oriented:** inheritance, encapsulation, polymorphism, abstraction સપોર્ટ કરે છે

### મેમરી ટ્રીક

“Simple Platform Object Security”

## પ્રશ્ન 2(ક) [7 ગુણ]

ઓવરરાઇડિંગ પદ્ધતિ શું છે? ઉદાહરણ સાથે સમજાવો.

### જવાબ

**Method Overriding:** ચાઇલ્ડ ક્લાસ પેરન્ટ ક્લાસની મેથડનું વિશિષ્ટ implementation પ્રદાન કરે છે.

#### Listing 4. Method Overriding

```

1 class Animal {
2     public void sound() {
3         System.out.println("Animal makes sound");
4     }
5 }
6
7 class Dog extends Animal {
8     @Override
9     public void sound() {
10        System.out.println("Dog barks");
11    }
12 }
13
14 public class Test {
15     public static void main(String[] args) {
16         Animal a = new Dog();
17         a.sound(); // Output: Dog barks
18     }
19 }
```

#### મુખ્ય મુદ્દાઓ:

- **Runtime Polymorphism:** ઓબ્જેક્ટ ટાઇપના આધારે મેથડ કોલ થાય છે
- **@Override:** મેથડ ઓવરરાઇડિંગ માટે annotation
- **Dynamic Binding:** રનટાઇમ પર મેથડ રિઝોલ્યુશન

### મેમરી ટ્રીક

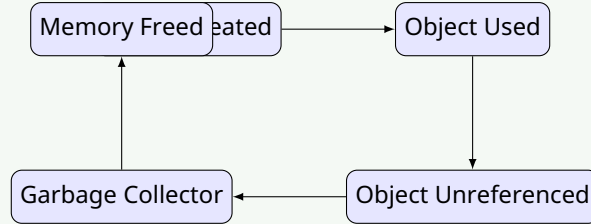
“Child Changes Parent Method”

## પ્રશ્ન 2(અ OR) [3 ગુણ]

જાવામાં Garbage collection સમજાવો.

જવાબ

**Garbage Collection:** ઓટોમેટિક મેમરી મેનેજમેન્ટ જે અનુપયોગી ઓબ્જેક્ટ્સને દૂર કરે છે.



મુખ્ય મુદ્દાઓ:

- **Automatic:** મેન્યુઅલ મેમરી deallocation નથી
- **Mark and Sweep:** અનુપયોગી ઓબ્જેક્ટ્સને ઓળખે અને દૂર કરે છે
- **Heap Memory:** heap memory area પર કામ કરે છે

મેમરી ટ્રીક

"Auto Clean Unused Objects"

## પ્રશ્ન 2(બ OR) [4 ગુણ]

static કીવર્ડ ઉદાહરણ સાથે સમજાવો.

જવાબ

**Static Keyword:** ઇન્સ્ટન્સને બદલે ક્લાસનું છે.

Listing 5. Static Example

```

1 class Student {
2     static String college = "GTU"; // Static variable
3     String name;
4
5     static void showCollege() { // Static method
6         System.out.println("College: " + college);
7     }
8 }
  
```

**Static Features:**

- **Memory:** ક્લાસ લોડિંગ ટાઇમે લોડ થાય છે
- **Access:** ઓબ્જેક્ટ વિના એક્સેસ કરી શકાય છે
- **Sharing:** બધા instances વચ્ચે શેર થાય છે

મેમરી ટ્રીક

"Class Level Memory Sharing"

## પ્રશ્ન 2(ક OR) [7 ગુણ]

કન્સ્ટ્રક્ટર શું છે? કોપી કન્સ્ટ્રક્ટરને ઉદાહરણ સાથે સમજાવો.

જવાબ

**Constructor:** ઓબ્જેક્ટ્સને initialize કરવા માટેની વિશેષ મેથડ.

Listing 6. Constructor Types

```

1 class Person {
2     String name;
3     int age;
4
5     // Default constructor
6     Person() {
7         name = "Unknown";
8         age = 0;
9     }
10
11    // Parameterized constructor
12    Person(String n, int a) {
13        name = n;
14        age = a;
15    }
16
17    // Copy constructor
18    Person(Person p) {
19        name = p.name;
20        age = p.age;
21    }
22 }
```

**Constructor Types:**

- **Default:** કોઈ પેરામીટર નથી
- **Parameterized:** પેરામીટર લે છે
- **Copy:** અસ્તિત્વમાં રહેલા ઓબ્જેક્ટમાંથી ઓબ્જેક્ટ બનાવે છે

મેમરી ટ્રીક

“Default Parameter Copy”

## પ્રશ્ન 3(અ) [3 ગુણ]

Super કીવર્ડ ઉદાહરણ સાથે સમજાવો.

જવાબ

**Super Keyword:** પેરન્ટ ક્લાસના મેમ્બર્સનો રેફરન્સ આપે છે.

Listing 7. Super Keyword

```

1 class Vehicle {
2     String brand = "Generic";
3 }
4
5 class Car extends Vehicle {
6     String brand = "Toyota";
7 }
```

```

8 void display() {
9     System.out.println("Child: " + brand);
10    System.out.println("Parent: " + super.brand);
11 }
12 }

```

#### Super ના ઉપયોગો:

- **Variables:** પેરન્ટ ક્લાસના વેરિયેબલ્સ એક્સેસ કરવા
- **Methods:** પેરન્ટ ક્લાસની મેથડ્સ કોલ કરવા
- **Constructor:** પેરન્ટ ક્લાસનું કન્સ્ટ્રક્ટર કોલ કરવા

#### મેમરી ટ્રીક

``Super Calls Parent"

## પ્રશ્ન 3(બ) [4 ગુણ]

ઇન્હેરિટન્સના વિવિધ પ્રકારોની યાદી આપો. મલ્ટિલેવલ ઇન્હેરિટન્સ સમજાવો.

#### જવાબ

#### Inheritance Types:

કોષ્ટક 3. Inheritance Types

પ્રકાર	વર્ણન
Single	એક પેરન્ટ, એક ચાઇલ્ડ
Multilevel	ઇન્હેરિટન્સની ચેઇન
Hierarchical	એક પેરન્ટ, અનેક ચાઇલ્ડ
Multiple	અનેક પેરન્ટ (ઇન્ટરફેસ દ્વારા)

#### Multilevel Inheritance:

Listing 8. Multilevel Inheritance

```

1 class Animal {
2     void eat() { System.out.println("Eating"); }
3 }
4
5 class Mammal extends Animal {
6     void breathe() { System.out.println("Breathing"); }
7 }
8
9 class Dog extends Mammal {
10    void bark() { System.out.println("Barking"); }
11 }

```

#### મેમરી ટ્રીક

``Single Multi Hierarchical Multiple"

## પ્રશ્ન 3(ક) [7 ગુણ]

ઇન્ટરફેસ શું છે? ઉદાહરણ સાથે મલ્ટીપલ ઇન્હેરિટન્સ સમજાવો.

## જવાબ

**Interface:** કોન્ટ્રાક્ટ જે વ્યાખ્યાયિત કરે છે કે ક્લાસ શું કરવું જોઈએ, કેવી રીતે નહીં.

Listing 9. Multiple Inheritance with Interface

```

1 interface Flyable {
2     void fly();
3 }
4
5 interface Swimmable {
6     void swim();
7 }
8
9 class Duck implements Flyable, Swimmable {
10     public void fly() {
11         System.out.println("Duck is flying");
12     }
13
14     public void swim() {
15         System.out.println("Duck is swimming");
16     }
17 }

```

**Interface Features:**

- **Multiple Inheritance:** ક્લાસ એકથી વધુ ઇન્ટરફેસને implement કરી શકે છે
- **Abstract Methods:** બધી મેથડ્સ બાય ડિફોલ્ટ abstract હોય છે
- **Constants:** બધા વેરિએબલ્સ public, static, final હોય છે

## મેમરી ટ્રીક

“Multiple Abstract Constants”

## પ્રશ્ન 3(અ OR) [3 ગુણ]

final કીવર્ડ ઉદાહરણ સાથે સમજાવો.

## જવાબ

**Final Keyword:** ફેરફાર, ઇનહેરિટેન્સ અથવા ઓવરરાઇડિંગને પ્રતિબંધિત કરે છે.

Listing 10. Final Keyword

```

1 final class Math {    // ઇનહેરિટ થઈ શકતું નથી
2     final int PI = 3.14; // બદલી શકાતું નથી
3
4     final void calculate() { // ઓવરરાઇડ થઈ શકતું નથી
5         System.out.println("Calculating");
6     }
7 }

```

**Final ના ઉપયોગો:**

- **Class:** એક્સટેન્ડ કરી શકાતું નથી
- **Method:** ઓવરરાઇડ કરી શકાતું નથી
- **Variable:** ફરીથી assign કરી શકાતું નથી

મેમરી ટ્રીક

"Final Stops Changes"

## પ્રશ્ન 3(બ OR) [4 ગુણ]

જાવામાં વિવિધ access controls સમજાવો.

જવાબ

Access Modifiers:

કોષ્ટક 4. Access Modifiers

Modifier	Same Class	Same Package	Subclass	Diff Package
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗

મેમરી ટ્રીક

"Public Protected Default Private"

## પ્રશ્ન 3(ક OR) [7 ગુણ]

પેકેજ શું છે? પેકેજ બનાવવા માટેના પગલાં લખો અને તેનું ઉદાહરણ આપો.

જવાબ

**Package:** સંબંધિત ક્લાસ અને ઇન્ટરફેસનું જૂથ.**પેકેજ બનાવવાના પગલાં:**

1. **Declare:** ટોચ પર package વિધાન વાપરો
2. **Compile:** javac -d . ClassName.java
3. **Run:** java packagename.ClassName

Listing 11. Package Example

```

1 // File: mypack/Calculator.java
2 package mypack;
3
4 public class Calculator {
5     public int add(int a, int b) {
6         return a + b;
7     }
8 }
9
10 // File: Test.java
11 import mypack.Calculator;
12
13 public class Test {
14     public static void main(String[] args) {
15         Calculator calc = new Calculator();
16         System.out.println(calc.add(5, 3));
17     }

```

18 }

**Package ના ફાયદા:**

- **Organization:** સંબંધિત ક્લાસને જૂથબદ્ધ કરે છે
- **Access Control:** પેકેજ-લેવેલ સુરક્ષા
- **Namespace:** નામના સંઘર્ષ (naming conflicts) ને ટાળે છે

**મેમરી ટ્રીક**

``Declare Compile Run"

## પ્રશ્ન 4(અ) [3 ગુણ]

યોગ્ય ઉદાહરણ સાથે થ્રેડ પ્રાયોરિટીઝ સમજાવો.

**જવાબ****Thread Priority:** થ્રેડ એક્ઝિક્યુશનનો ક્રમ નક્કી કરે છે (1-10 સ્કેલ).**Listing 12.** Thread Priority

```

1 class MyThread extends Thread {
2     public void run() {
3         System.out.println(getName() + " Priority: " + getPriority());
4     }
5 }
6
7 public class ThreadPriorityExample {
8     public static void main(String[] args) {
9         MyThread t1 = new MyThread();
10        MyThread t2 = new MyThread();
11
12        t1.setPriority(Thread.MIN_PRIORITY); // 1
13        t2.setPriority(Thread.MAX_PRIORITY); // 10
14
15        t1.start();
16        t2.start();
17    }
18 }

```

**Priority Constants:**

- MIN\_PRIORITY: 1
- NORM\_PRIORITY: 5
- MAX\_PRIORITY: 10

**મેમરી ટ્રીક**

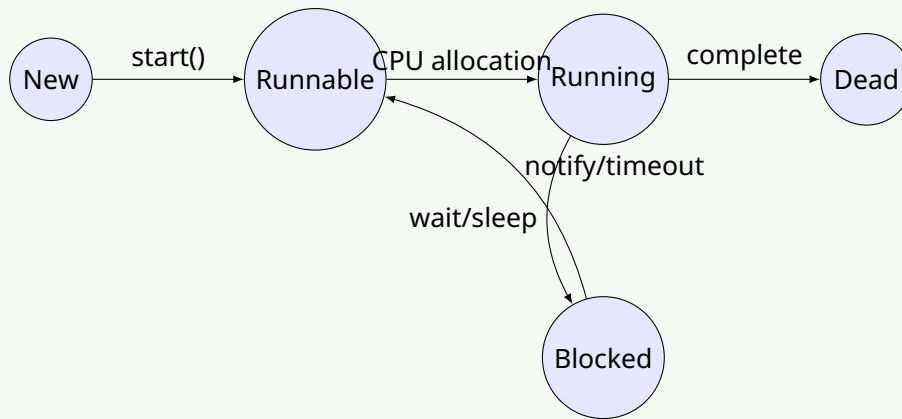
``Min Normal Max"

## પ્રશ્ન 4(બ) [4 ગુણ]

થ્રેડ શું છે? થ્રેડ લાઈફ સાયકલ સમજાવો.

## જવાબ

**Thread:** એકસાથે એક્ઝિક્યુશન માટે લાઇટવેઇટ પ્રોસેસ.



## Thread States:

- **New:** થ્રેડ બન્યો પણ શરૂ થયો નથી
- **Runnable:** રન થવા માટે તૈયાર
- **Running:** હાલમાં એક્ઝિક્યુટ થઈ રહ્યો છે
- **Blocked:** રિસોર્સ માટે રાહ જોઈ રહ્યો છે
- **Dead:** એક્ઝિક્યુશન પૂર્ણ થયું

## મેમરી ટ્રીક

"New Runnable Running Blocked Dead"

## પ્રશ્ન 4(ક) [7 ગુણ]

Runnable ઇન્ટરફેસનો અમલ કરીને મલ્ટીપલ થ્રેડ બનાવે તેવો જાવા પ્રોગ્રામ લખો.

## જવાબ

Listing 13. Multiple Threads

```

1 class MyRunnable implements Runnable {
2     private String threadName;
3
4     MyRunnable(String name) {
5         threadName = name;
6     }
7
8     public void run() {
9         for(int i = 1; i <= 5; i++) {
10             System.out.println(threadName + " - Count: " + i);
11             try {
12                 Thread.sleep(1000);
13             } catch (InterruptedException e) {
14                 e.printStackTrace();
15             }
16         }
17     }
18 }
19
20 public class MultipleThreads {

```

```

21 public static void main(String[] args) {
22     Thread t1 = new Thread(new MyRunnable("Thread-1"));
23     Thread t2 = new Thread(new MyRunnable("Thread-2"));
24     Thread t3 = new Thread(new MyRunnable("Thread-3"));
25
26     t1.start();
27     t2.start();
28     t3.start();
29 }
30 }

```

મુખ્ય મુદ્દાઓ:

- **Runnable Interface:** Thread ક્લાસને extend કરવા કરતા વધુ સારું
- **Thread.sleep():** થ્રેડ એક્ટિવિટીને થોભાવે છે
- **Multiple Threads:** એકસાથે રન થાય છે (conucently)

મેમરી ટ્રીક

“Implement Runnable Start Multiple”

## પ્રશ્ન 4(અ OR) [3 ગુણ]

ચાર અલગ અલગ ઇનબિલ્ટ એક્સેપ્શનની યાદી આપો. કોઈપણ એક સમજાવો.

જવાબ

Inbuilt Exceptions:

- **NullPointerException:** null ઓબ્જેક્ટ એક્સેસ કરવું
- **ArrayIndexOutOfBoundsException:** અમાન્ય એરે ઇન્ડેક્સ
- **ArithmeticException:** શૂન્ય વડે ભાગાકાર
- **NumberFormatException:** અમાન્ય નંબર ફોર્મેટ

**ArithmeticException:** જ્યારે એરિથમેટિક ઓપરેશન નિષ્ફળ જાય ત્યારે throw થાય છે.

Listing 14. ArithmeticException

```

1 int result = 10 / 0; // Throws ArithmeticException

```

મેમરી ટ્રીક

“Null Array Arithmetic Number”

## પ્રશ્ન 4(બ OR) [4 ગુણ]

Try અને Catch યોગ્ય ઉદાહરણ સાથે સમજાવો.

જવાબ

**Try-Catch:** એક્સેપ્શન હેન્ડલિંગ મિકેનિઝમ.

Listing 15. Try Catch Example

```

1 public class TryCatchExample {
2     public static void main(String[] args) {
3         try {

```

```

4      int[] arr = {1, 2, 3};
5      System.out.println(arr[5]); // Index out of bounds
6  }
7  catch(ArrayIndexOutOfBoundsException e) {
8      System.out.println("Array index error: " + e.getMessage());
9  }
10 finally {
11     System.out.println("Always executed");
12 }
13 }
14 }

```

**Exception Handling Flow:**

- **Try:** કોડ જે એક્સેપ્શન throw કરી શકે છે
- **Catch:** થોક્કસ એક્સેપ્શન હેન્ડલ કરે છે
- **Finally:** હંમેશા એક્ઝિક્યુટ થાય છે

**મેમરી ટ્રીક**

“Try Catch Finally”

**પ્રશ્ન 4(ક OR) [7 ગુણ]**

Exception શું છે? Arithmetic Exception નો ઉપયોગ દર્શાવતો પ્રોગ્રામ લખો.

**જવાબ**

**Exception:** રનટાઇમ એરર જે સામાન્ય પ્રોગ્રામ ફ્લોને વિક્ષેપિત કરે છે.

**Listing 16.** ArithmeticException Example

```

1  public class ArithmeticExceptionExample {
2      public static void main(String[] args) {
3          Scanner sc = new Scanner(System.in);
4
5          try {
6              System.out.print("Enter first number: ");
7              int num1 = sc.nextInt();
8
9              System.out.print("Enter second number: ");
10             int num2 = sc.nextInt();
11
12             int result = num1 / num2;
13             System.out.println("Result: " + result);
14         }
15         catch(ArithmeticException e) {
16             System.out.println("Error: Cannot divide by zero!");
17         }
18         catch(Exception e) {
19             System.out.println("General error: " + e.getMessage());
20         }
21         finally {
22             sc.close();
23         }
24     }
25 }

```

**Exception Types:**

- **Checked:** Compile-time exceptions

- **Unchecked:** Runtime exceptions
- **Error:** System-level problems

### મેમરી ટ્રીક

“Runtime Error Disrupts Flow”

## પ્રશ્ન 5(અ) [3 ગુણ]

Java માં `ArrayIndexOutOfBoundsException` Exception ઉદાહરણ સાથે સમજાવો.

### જવાબ

**`ArrayIndexOutOfBoundsException`:** જ્યારે અમાન્ય એરે ઇન્ડેક્સ એક્સેસ કરવામાં આવે ત્યારે throw થાય છે.

Listing 17. `ArrayIndexOutOfBoundsException`

```

1 public class ArrayIndexExample {
2     public static void main(String[] args) {
3         int[] numbers = {10, 20, 30};
4
5         try {
6             System.out.println(numbers[5]); // Invalid index
7         }
8         catch (ArrayIndexOutOfBoundsException e) {
9             System.out.println("Invalid array index: " + e.getMessage());
10        }
11    }
12 }
```

મુખ્ય મુદ્દાઓ:

- **Valid Range:** 0 to array.length-1
- **Negative Index:** આ પણ એક્સેપ્શન throw કરે છે
- **Runtime Exception:** Unchecked exception

### મેમરી ટ્રીક

“Array Index Range Check”

## પ્રશ્ન 5(બ) [4 ગુણ]

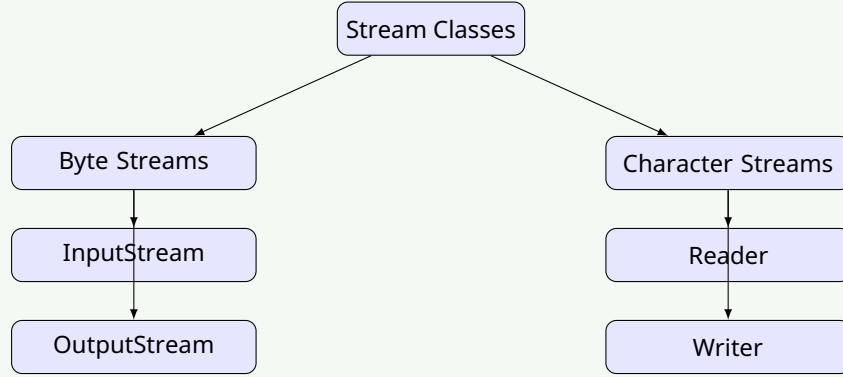
સ્ટ્રીમ ક્લાસીસના બેઝિક્સ સમજાવો.

### જવાબ

**Stream Classes:** ઇનપુટ/આઉટપુટ ઓપરેશન્સ હેન્ડલ કરે છે.

કોષ્ટક 5. Stream Classes

Stream Type	Classes
Byte Streams	InputStream, OutputStream
Character Streams	Reader, Writer
File Streams	FileInputStream, FileOutputStream
Buffered Streams	BufferedReader, BufferedWriter

**Stream Features:**

- **Sequential:** ડેટા ક્રમમાં વહે છે
- **One Direction:** કાં તો ઇનપુટ અથવા આઉટપુટ
- **Automatic:** નીચલા સ્તરની વિગતો (low-level details) હેન્ડલ કરે છે

**મેમરી ટ્રીક**

“Byte Character File Buffered”

**પ્રશ્ન 5(ક) [7 ગુણ]**

ટેક્સ્ટ ફાઇલ બનાવવા માટે જાવા પ્રોગ્રામ લખો અને ટેક્સ્ટ ફાઇલ પર રીડ ઓપરેશન કરો.

**જવાબ****Listing 18. File Create and Read**

```

1 import java.io.*;
2
3 public class FileReadExample {
4     public static void main(String[] args) {
5         // Create and write to file
6         try {
7             FileWriter writer = new FileWriter("sample.txt");
8             writer.write("Hello World!\n");
9             writer.write("Java File Handling\n");
10            writer.write("GTU Exam 2024");
11            writer.close();
12            System.out.println("File created successfully");
13        }
14        catch(IOException e) {
15            System.out.println("Error creating file: " + e.getMessage());
16        }
17
18        // Read from file
19        try {

```

```

20     BufferedReader reader = new BufferedReader(new FileReader("sample.txt"));
21     String line;
22
23     System.out.println("\nFile contents:");
24     while((line = reader.readLine()) != null) {
25         System.out.println(line);
26     }
27     reader.close();
28 }
29 catch(IOException e) {
30     System.out.println("Error reading file: " + e.getMessage());
31 }
32 }
33 }

```

મુખ્ય મુદ્દાઓ:

- **FileWriter:** ફાઇલ બનાવે અને લખે છે
- **BufferedReader:** કાર્યક્ષમ વાંચન
- **Exception Handling:** IOException handle કરો

મેમરી ટ્રીક

“Create Write Read Close”

## પ્રશ્ન 5(અ OR) [3 ગુણ]

Java માં Divide by Zero Exception ને ઉદાહરણ સાથે સમજાવો.

જવાબ

**ArithmeticException:** શૂન્યથી ભાગાકાર ઓપરેશન દરમિયાન throw થાય છે.

Listing 19. Divide by Zero

```

1 public class DivideByZeroExample {
2     public static void main(String[] args) {
3         try {
4             int a = 10;
5             int b = 0;
6             int result = a / b; // Throws ArithmeticException
7             System.out.println("Result: " + result);
8         }
9         catch(ArithmeticException e) {
10             System.out.println("Cannot divide by zero: " + e.getMessage());
11         }
12     }
13 }

```

મુખ્ય મુદ્દાઓ:

- **Integer Division:** માત્ર integer division by zero exception throw કરે છે
- **Floating Point:** floating point division માટે Infinity return કરે છે
- **Runtime Exception:** Unchecked exception

મેમરી ટ્રીક

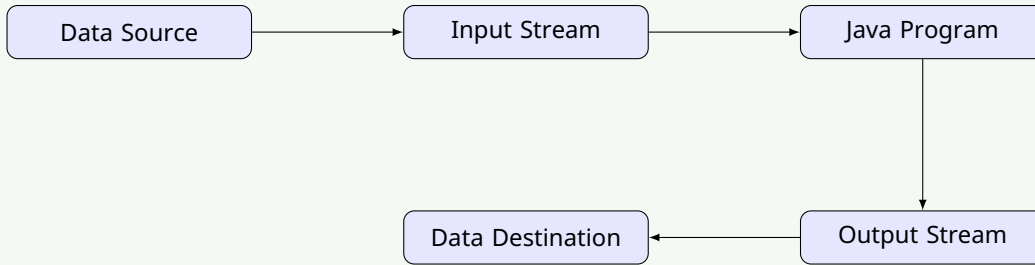
“Zero Division Arithmetic Error”

## પ્રશ્ન 5(બ OR) [4 ગુણ]

java I/O પ્રક્રિયા સમજાવો.

જવાબ

Java I/O Process: ડેટા વાંચવા અને લખવાની પદ્ધતિ.



I/O Components:

- **Stream:** ડેટાનો ક્રમ
- **Buffer:** કાર્યક્ષમતા માટે અસ્થાયી સ્ટોરેજ
- **File:** સ્થાયી સ્ટોરેજ
- **Network:** દૂરસ્થ ડેટા ટ્રાન્સફર

I/O Types:

- **Byte-oriented:** કાચો ડેટા (images, videos)
- **Character-oriented:** ટેક્સ્ટ ડેટા
- **Synchronous:** Blocking operations
- **Asynchronous:** Non-blocking operations

મેમરી ટ્રીક

“Stream Buffer File Network”

## પ્રશ્ન 5(ક OR) [7 ગુણ]

ટેક્સ્ટ ફાઇલ બનાવવા માટે જાવા પ્રોગ્રામ લખો અને ટેક્સ્ટ ફાઇલ પર રાઇટ ઓપરેશન કરો.

જવાબ

Listing 20. File Write Example

```

1 import java.io.*;
2 import java.util.Scanner;
3
4 public class FileWriteExample {
5     public static void main(String[] args) {
6         Scanner sc = new Scanner(System.in);
7
8         try {
9             // Create file with FileWriter
10            FileWriter writer = new FileWriter("student.txt");
11
12            System.out.println("વહિયાર્થીની વગિતો દાખલ કરો:");
13            System.out.print("નામ: ");
14            String name = sc.nextLine();
15
16            System.out.print("રોલ નંબર: ");
  
```

```

17     String rollNo = sc.nextLine();
18
19     System.out.print("શાખા: ");
20     String branch = sc.nextLine();
21
22     // Write data to file
23     writer.write("Student Information\n");
24     writer.write("=====\n");
25     writer.write("નામ: " + name + "\n");
26     writer.write("રોલ નંબર: " + rollNo + "\n");
27     writer.write("શાખા: " + branch + "\n");
28     writer.write("તારીખ: " + new java.util.Date() + "\n");
29
30     writer.close();
31     System.out.println("\nડેટા સફળતાપૂર્વક ફાઇલમાં લખાયો!");
32
33 }
34 catch(IOException e) {
35     System.out.println("ફાઇલમાં લખવામાં એરર: " + e.getMessage());
36 }
37 finally {
38     sc.close();
39 }
40 }
41 }

```

#### મુખ્ય મુદ્દાઓ:

- **FileWriter**: ફાઇલમાં character data લખે છે
- **BufferedWriter**: મોટા ડેટા માટે વધુ કાર્યક્ષમ
- **Auto-close**: automatic closing માટે try-with-resources વાપરો

#### મેમરી ટ્રીક

``Create Write Close Handle"