જાવા પ્રોગ્રામિંગ (4343203) – સમર 2025 સોલ્યુશન

Milav Dabgar

May 13, 2025

પ્રશ્ન 1(અ) [3 ગુણ]

Java માં identifier ને નામ આપવાના નિયમોની યાદી માન્ય અને અમાન્ય ઉદાહરણો સાથે બનાવો.

જવાબ

Java Identifier ના નિયમો:

કોષ્ટક 1. Identifier નિયમો

નિયમ	વર્ણન	માન્ય ઉદાહરણ	અમાન્ય ઉદાહરણ
શરૂઆતી અક્ષર	letter, underscore અથવા dollar sign થી શરૂ	name, _value, $cost	2name, #id
પછીના અક્ષરો	letters, digits, underscore, dollar હોઈ શકે	student123, user_name	my-var, class@
Keyword પ્રતિબંધ	Java reserved words વાપરી શકાતા નથી	myClass, userName	class, int
Case Sensitivity	Identifier case-sensitive છે	Name $\neq$ name	--
લંબાઈ	કોઈ લંબાઈની મર્યાદા નથી	verylongvariablename	--

મેમરી ટ્રીક

``Letters First, Keywords Never, Case Counts''

પ્રશ્ન 1(બ) [4 ગુણ]

Java ના operator ની યાદી બનાવો. Arithmetic અને Logical operator ને સમજાવો.

જવાબ

Java Operator ના પ્રકારો:

કોષ્ટક 2. Operator ના પ્રકારો

Operator નો પ્રકાર	ઉદાહરણો
Arithmetic	+, -, *, /, %
Relational	==, !=, <, >, <=, >=
Logical	&&, ||, !
Assignment	=, +=, -=, *=, /=
Unary	++, --, +, -, !
Bitwise	&, |, ~, <<, >>
Ternary	condition ? value1 : value2

Arithmetic Operators:

- **Addition (+)**: બે operand નો સરવાળો કરે છે
- **Subtraction (-)**: પહેલામાંથી બીજો બાદ કરે છે
- **Multiplication (*)**: બે operand નો ગુણાકાર કરે છે
- **Division (/)**: પહેલાને બીજા વડે ભાગ આપે છે
- **Modulus (%)**: ભાગાકારનો શેષ આપે છે

Logical Operators:

- **AND (&&)**: બંને શરતો સાચી હોય તો true આપે છે
- **OR (||)**: ઓછામાં ઓછી એક શરત સાચી હોય તો true આપે છે
- **NOT (!)**: logical state ને ઉલટાવે છે

મેમરી ટ્રીક

``Add Subtract Multiply Divide Remainder, And Or Not''

પ્રશ્ન 1(ક) [7 ગુણ]

3 આંકડાની સંખ્યાને વિપરીત કરવા માટે Java પ્રોગ્રામ લખો. ઉદાહરણ તરીકે, સંખ્યા 653 છે તો તેની વિપરીત સંખ્યા 356 છે.

જવાબ

Listing 1. Reverse Number

```java
import java.util.Scanner;

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

        System.out.print("3 આંકડાની સંખ્યા દાખલ કરો: ");
        int num = sc.nextInt();

        int reverse = 0;
        int temp = num;

        while (temp > 0) {
            reverse = reverse * 10 + temp % 10;
            temp = temp / 10;
        }

        System.out.println("મૂળ સંખ્યા: " + num);
        System.out.println("વિપરીત સંખ્યા: " + reverse);
    }
}
```

Algorithm:

- **છેલ્લો આંકડો કાઢો**: Modulus operator (%) નો ઉપયોગ કરો
- **વિપરીત સંખ્યા બનાવો**: 10 થી ગુણો અને આંકડો ઉમેરો
- **છેલ્લો આંકડો દૂર કરો**: Integer division (/) નો ઉપયોગ કરો
- **પુનરાવર્તન કરો**: મૂળ સંખ્યા 0 થાય ત્યાં સુધી

મેમરી ટ્રીક

``Extract, Build, Remove, Repeat''

પ્રશ્ન 1(ક OR) [7 ગુણ]

3*3 મેટ્રિક્સનો સરવાળો કરવા માટેનો Java પ્રોગ્રામ લખો.

જવાબ

Listing 2. Matrix Addition

```
import java.util.Scanner;

public class MatrixAddition {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] matrix1 = new int[3][3];
        int[][] matrix2 = new int[3][3];
        int[][] result = new int[3][3];

        // પહેલું મેટ્રિક્સ દાખલ કરો
        System.out.println("પહેલું મેટ્રિક્સ દાખલ કરો:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                matrix1[i][j] = sc.nextInt();
            }
        }

        // બીજું મેટ્રિક્સ દાખલ કરો
        System.out.println("બીજું મેટ્રિક્સ દાખલ કરો:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                matrix2[i][j] = sc.nextInt();
            }
        }

        // મેટ્રિક્સનો સરવાળો
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                result[i][j] = matrix1[i][j] + matrix2[i][j];
            }
        }

        // પરિણામ દર્શાવો
        System.out.println("મેટ્રિક્સનો સરવાળો:");
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                System.out.print(result[i][j] + " ");
            }
            System.out.println();
        }
    }
}
```

Matrix Addition ના પગલાં:

- **Array બનાવો**: ત્રણ 3×3 integer array
- **મેટ્રિક્સ દાખલ કરો**: બંને મેટ્રિક્સ માટે values વાંચો
- **સંબંધિત elements ઉમેરો**: result[i][j] = matrix1[i][j] + matrix2[i][j]
- **પરિણામ દર્શાવો**: સરવાળાનું મેટ્રિક્સ print કરો

મેમરી ટ્રીક

``Create, Input, Add, Display''

પ્રશ્ન 2(અ) [3 ગુણ]

પેરામીટરાઇઝ્ડ કન્સ્ટ્રક્ટર નો ઉપયોગ દર્શાવતો Java પ્રોગ્રામ લખો.

જવાબ

Listing 3. Parameterized Constructor

```
class Student {
    private String name;
    private int rollNo;

    // Parameterized Constructor
    public Student(String name, int rollNo) {
        this.name = name;
        this.rollNo = rollNo;
    }

    public void display() {
        System.out.println("નામ: " + name);
        System.out.println("રોલ નંબર: " + rollNo);
    }
}

public class ParameterizedConstructor {
    public static void main(String[] args) {
        Student s1 = new Student("જોન", 101);
        s1.display();
    }
}
```

Parameterized Constructor ની વિશેષતાઓ:

- **પેરામીટર લે છે**: Object બનાવતી વખતે values સ્વીકારે છે
- **Instance variables initialize કરે છે**: Object ની state સેટ કરે છે
- **Class જેવું જ નામ**: Constructor નું નામ class સાથે મેળ ખાય છે
- **Return type નથી**: Constructor નો return type હોતો નથી

મેમરી ટ્રીક

``Parameters Initialize Same-name No-return''

પ્રશ્ન 2(બ) [4 ગુણ]

ઉદાહરણ સાથે નીચેના શબ્દોની બેસીક સીન્ટેક્સ આપો: (1) ક્લાસ બનાવવા માટે, (2) ઓબ્જેક્ટ બનાવવા માટે, (3) મેથડને વ્યાખ્યાયિત કરવા માટે, (4) ચલ(વેરીએબલ)ને ડિક્લેર કરવા માટે.

જવાબ

Java Basic Syntax:

કોષ્ટક 3. Basic Syntax

ઘટક	Syntax	ઉદાહરણ
Class બનાવવું	class ClassName { }	class Car { }
Object બનાવવું	ClassName obj = new ClassName();	Car c = new Car();
Method વ્યાખ્યા	returnType method(params) { }	void start() { }
Variable declaration	dataType varName;	int age;

Listing 4. Syntax Example

```
class Car {                    // Class બનાવવું
   int speed;                  // Variable Declaration

   public void accelerate() {      // Method વ્યાખ્યા
      speed += 10;
   }
}

public class Main {
   public static void main(String[] args) {
      Car myCar = new Car();       // Object બનાવવું
   }
}
```

મેમરી ટ્રીક

``Class Object Method Variable - COMV''

પ્રશ્ન 2(ક) [7 ગુણ]

Java માં એક પ્રોગ્રામ લખો જેમાં Student નામનો Class છે. જેમાં enrollmentNo અને name નામ ના બે ઇન્સ્ટન્ટ વેરીએબલ હોય. Student Class ના 3 ઓબ્જેક્ટ main મેથડમાં બનાવો અને Student's ના નામ દર્શાવો.

જવાબ

Listing 5. Student Class Demo

```
class Student {
   String enrollmentNo;
   String name;

   // Student data initialize કરવા માટે constructor
   public Student(String enrollmentNo, String name) {
      this.enrollmentNo = enrollmentNo;
      this.name = name;
   }

   // Student નું નામ display કરવા માટે method
   public void displayName() {
      System.out.println("વિદ્યાર્થીનું નામ: " + name);
   }
}

public class StudentDemo {
   public static void main(String[] args) {
      // Student class ના 3 objects બનાવવા
      Student s1 = new Student("CS001", "અલસિ");
```

```
        Student s2 = new Student("CS002", "બોબ");
        Student s3 = new Student("CS003", "ચાર્લી");

        // વિદ્યાર્થીઓના નામ દર્શાવવા
        s1.displayName();
        s2.displayName();
        s3.displayName();
    }
}
```

પ્રોગ્રામની રચના:
- **Class વ્યાખ્યા**: Instance variables સાથે Student class
- **Constructor**: enrollmentNo અને name initialize કરવા માટે
- **Method**: displayName() વિદ્યાર્થીનું નામ બતાવવા માટે
- **Object બનાવવું**: main method માં ત્રણ Student objects
- **Method calling**: displayName() વાપરીને નામ દર્શાવવા

મેમરી ટ્રીક

``Define Initialize Display Create Call''

પ્રશ્ન 2(અ OR) [3 ગુણ]

ડિફોલ્ટ કન્સ્ટ્રક્ટર નો ઉપયોગ દર્શાવતો Java પ્રોગ્રામ લખો.

જવાબ

Listing 6. Default Constructor

```
class Rectangle {
    int length;
    int width;

    // Default Constructor
    public Rectangle() {
        length = 5;
        width = 3;
        System.out.println("Default constructor કોલ થયો");
    }

    public void displayArea() {
        System.out.println("વિસ્તાર: " + (length * width));
    }
}

public class DefaultConstructor {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle();
        r1.displayArea();
    }
}
```

Default Constructor ની વિશેષતાઓ:
- **કોઈ પેરામીટર નથી**: કોઈ arguments લેતો નથી
- **Default values**: Instance variables માટે default values સેટ કરે છે
- **આપોઆપ કોલ**: Object બનાવતી વખતે કોલ થાય છે
- **Class જેવું જ નામ**: Constructor નું નામ class સાથે મેળ ખાય છે

મેમરી ટ્રીક

``No-parameters Default Automatic Same-name''

પ્રશ્ન 2(બ OR) [4 ગુણ]

પ્રોસિજર-ઓરિએન્ટેડ પ્રોગ્રામિંગ અને ઓબ્જેક્ટ-ઓરિએન્ટેડ પ્રોગ્રામિંગ વચ્ચે ચાર તફાવત આપો.

જવાબ

POP vs OOP તુલના:

કોષ્ટક 4. POP vs OOP

પાસું	Procedure Oriented Programming	Object-Oriented Programming
અભિગમ	Top-down approach	Bottom-up approach
ધ્યાન	Functions અને procedures પર	Objects અને classes પર
Data સુરક્ષા	Data hiding નથી, global access	Data encapsulation અને hiding
સમસ્યા ઉકેલ	Functions માં વહેંચવું	Objects માં વહેંચવું
Code પુનઃઉપયોગ	મર્યાદિત પુનઃઉપયોગ	Inheritance દ્વારા ઉચ્ચ પુનઃઉપયોગ
જાળવણી	જાળવવું મુશ્કેલ	જાળવવું અને સુધારવું સરળ

મેમરી ટ્રીક

``Structure Security Reusability Maintenance''

પ્રશ્ન 2(ક OR) [7 ગુણ]

Java માં એક પ્રોગ્રામ લખો જેમાં Shape નામની Class હોય જે 2 ઓવરલોડ મેથડ area (ફ્લોટ radius) અને area (ફ્લોટ length, ફ્લોટ width) ધરાવતો હોય. ઓવરલોડ પદ્ધતિઓનો ઉપયોગ કરીને વર્તુળ અને લંબચોરસનો area (વિસ્તાર) દર્શાવો.

જવાબ

Listing 7. Method Overloading

```
class Shape {
    // વર્તુળનો વિસ્તાર કેલ્ક્યુલેટ કરવા માટે method
    public void area(float radius) {
        float circleArea = 3.14f * radius * radius;
        System.out.println("વર્તુળનો વિસ્તાર: " + circleArea);
    }

    // લંબચોરસનો વિસ્તાર કેલ્ક્યુલેટ કરવા માટે overloaded method
    public void area(float length, float width) {
        float rectangleArea = length * width;
        System.out.println("લંબચોરસનો વિસ્તાર: " + rectangleArea);
    }
}

public class MethodOverloading {
    public static void main(String[] args) {
        Shape shape = new Shape();

```

```
        // radius 5 સાથે વર્તુળનો વિસ્તાર કેલ્ક્યુલેટ કરો
        shape.area(5.0f);

        // length 4 અને width 6 સાથે લંબચોરસનો વિસ્તાર કેલ્ક્યુલેટ કરો
        shape.area(4.0f, 6.0f);
    }
}
```

Method Overloading ની વિભાવનાઓ:

- **સમાન method નામ**: બંને methods નું નામ ``area''
- **ને પેરામીટર**: એક radius લે છે, બીજો length અને width લે છે
- **Compile-time polymorphism**: Method compile time પર પસંદ થાય છે

મેમરી ટ્રીક

``Same-name Different-parameters Compile-time Parameter-differentiation''

પ્રશ્ન 3(અ) [3 ગુણ]

સિંગલ ઇનહેરીટેન્સ દર્શાવવા માટે Java પ્રોગ્રામ લખો.

જવાબ

Listing 8. Single Inheritance

```
// Parent class
class Animal {
    public void eat() {
        System.out.println("પ્રાણી ખાય છે");
    }

    public void sleep() {
        System.out.println("પ્રાણી સૂય છે");
    }
}

// Animal માંથી inherit કરતો Child class
class Dog extends Animal {
    public void bark() {
        System.out.println("કૂતરો ભસે છે");
    }
}

public class SingleInheritance {
    public static void main(String[] args) {
        Dog dog = new Dog();

        // Animal class માંથી inherited methods
        dog.eat();
        dog.sleep();

        // Dog class નો પોતાનો method
        dog.bark();
    }
}
```

મેમરી ટ્રીક

``One-parent Extends Method IS-A''

પ્રશ્ન 3(બ) [4 ગુણ]

Java માં એબ્સ્ટ્રેક્ટ ક્લાસને ઉદાહરણ સાથે વ્યાખ્યાયિત કરો.

જવાબ

Abstract Class વ્યાખ્યા: Abstract class એ એવો class છે જેનો instantiate કરી શકાતો નથી અને તેમાં abstract methods (implementation વિનાના methods) હોઈ શકે છે.

Listing 9. Abstract Class

```
abstract class Vehicle {
    String brand;
    public void displayBrand() {
        System.out.println("બ્રાન્ડ: " + brand);
    }
    public abstract void start();
    public abstract void stop();
}

class Car extends Vehicle {
    public Car(String brand) {
        this.brand = brand;
    }
    public void start() {
        System.out.println("કાર ચાવી વડે સ્ટાર્ટ થયો");
    }
    public void stop() {
        System.out.println("કાર બ્રેક વડે બંધ થયો");
    }
}

public class AbstractDemo {
    public static void main(String[] args) {
        Car car = new Car("ટોયોટા");
        car.displayBrand();
        car.start();
        car.stop();
    }
}
```

મેમરી ટ્રીક

``Cannot-instantiate Abstract-methods Concrete-methods Must-extend''

પ્રશ્ન 3(ક) [7 ગુણ]

ઇન્ટરફેસનો ઉપયોગ કરીને મલ્ટીપલ ઇનહેરીટન્સને ઇમ્પ્લીમેંટ કરવા માટે Java માં પ્રોગ્રામ લખો.

જવાબ

Listing 10. Interface Multiple Inheritance

```
interface Flyable {
    void fly();
}

interface Swimmable {
    void swim();
}

class Duck implements Flyable, Swimmable {
    private String name;

    public Duck(String name) {
        this.name = name;
    }

    public void fly() {
        System.out.println(name + " આકાશમાં ઉડે છે");
    }

    public void swim() {
        System.out.println(name + " પાણીમાં સ્વમિ કરે છે");
    }

    public void walk() {
        System.out.println(name + " જમીન પર ચાલે છે");
    }
}

public class MultipleInheritance {
    public static void main(String[] args) {
        Duck duck = new Duck("ડોનાલ્ડ");
        duck.fly();
        duck.swim();
        duck.walk();
    }
}
```

મેમરી ટ્રીક

``Multiple-interfaces Implements Must-implement Solves-diamond''

પ્રશ્ન 3(અ OR) [3 ગુણ]

મલ્ટીલેવલ ઇનહેરીટેન્સ દર્શાવવા માટે Java પ્રોગ્રામ લખો.

જવાબ

Listing 11. Multilevel Inheritance

```
class Animal {
    public void breathe() {
        System.out.println("પ્રાણી શ્વાસ લે છે");
    }
}
```

```
class Mammal extends Animal {
    public void giveBirth() {
        System.out.println("સ્તનધારી બચ્ચાંને જન્મ આપે છે");
    }
}

class Dog extends Mammal {
    public void bark() {
        System.out.println("કૂતરો ભસે છે");
    }
}

public class MultilevelInheritance {
    public static void main(String[] args) {
        Dog dog = new Dog();
        dog.breathe();
        dog.giveBirth();
        dog.bark();
    }
}
```

મેમરી ટ્રીક

``Chain Multiple Transitive Extends''

પ્રશ્ન 3(બ OR) [4 ગુણ]

પેકેજ વ્યાખ્યાયિત કરો અને ઉદાહરણ સાથે પેકેજ બનાવવા માટે સીન્ટેક્સ લખો.

જવાબ

Package વ્યાખ્યા: Package એ namespace છે જે સંબંધિત classes અને interfaces ને organize કરે છે.
Package Syntax: package packageName;

Listing 12. Package Implementation

```
package mypackage;

public class Calculator {
    public int add(int a, int b) {
        return a + b;
    }
}

// બીજી ફાઈલમાં
import mypackage.Calculator;

public class TestCalculator {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        System.out.println("સરવાળો: " + calc.add(10, 5));
    }
}
```

મેમરી ટ્રીક

``Namespace Access Organization Reusability''

પ્રશ્ન 3(ક OR) [7 ગુણ]

મેથડ ઓવરરાઇડિંગ દર્શાવવા માટે Java પ્રોગ્રામ લખો.

જવાબ

Listing 13. Method Overriding

```
class Animal {
    public void makeSound() {
        System.out.println("પ્રાણી અવાજ કરે છે");
    }
    public void move() {
        System.out.println("પ્રાણી ફરે છે");
    }
}

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("કૂતરો ભસે છે: ભવ! ભવ!");
    }
    @Override
    public void move() {
        System.out.println("કૂતરો ચાર પગ પર દોડે છે");
    }
}

class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("બિલાડી મ્યાઉ કરે છે: મ્યાઉ! મ્યાઉ!");
    }
    @Override
    public void move() {
        System.out.println("બિલાડી શાંતિથી ચાલે છે");
    }
}

public class MethodOverriding {
    public static void main(String[] args) {
        Animal animal = new Dog();
        animal.makeSound();
        animal.move();

        animal = new Cat();
        animal.makeSound();
        animal.move();
    }
}
```

મેમરી ટ્રીક

``Same-signature Runtime Override IS-A''

પ્રશ્ન 4(અ) [3 ગુણ]

Java માં વિવિધ પ્રકારની એરરની યાદી બનાવો અને સમજાવો.

જવાબ

Java Error ના પ્રકારો:

કોષ્ટક 5. Error Types

Error નો પ્રકાર	વર્ણન	ઉદાહરણ
Compile-time	Compilation દરમિયાન શોધાય છે	Syntax errors, missing semicolons
Runtime	Program execution દરમિયાન થાય છે	Division by zero, null pointer
Logical	Program ચાલે છે પણ ખોટું output આપે છે	ખોટા algorithm logic

મેમરી ટ્રીક

``Compile Runtime Logic - CRL''

પ્રશ્ન 4(બ) [4 ગુણ]

રેપર ક્લાસ શું છે? કોઈપણ બે રેપર ક્લાસનો ઉપયોગ સમજાવો.

જવાબ

Wrapper classes primitive data types ના object representation પ્રદાન કરે છે.

કોષ્ટક 6. Wrapper Classes

Primitive	Wrapper Class
int	Integer
double	Double
boolean	Boolean
char	Character

Listing 14. Wrapper Examples

```
// Integer Wrapper
int num = 100;
Integer intObj = Integer.valueOf(num); // Boxing
int value = intObj.intValue(); // Unboxing
int parsed = Integer.parseInt("123"); // Parsing

// Double Wrapper
Double doubleObj = Double.valueOf(45.67);
double val = Double.parseDouble("123.45");
```

મેમરી ટ્રીક

``Collections Null Utility Generics''

પ્રશ્ન 4(ક) [7 ગુણ]

બેંકિંગ એપ્લિકેશન વિકસાવવા માટે Java માં એક પ્રોગ્રામ લખો જેમાં વપરાશકર્તા 25000 રૂપિયા જમા કરે અને પછી 20000, 4000 રૂપિયા ઉપાડવાનું શરૂ કરે અને ત્યારબાદ જ્યારે વપરાશકર્તા રૂ. 2000 ઉપાડે તો પ્રોગ્રામ ``પૂરતું ભંડોળ નથી'' એક્સેપ્શન(exception) આપે.

જવાબ

Listing 15. Banking Application

```
class InsufficientFundException extends Exception {
    public InsufficientFundException(String message) {
        super(message);
    }
}

class BankAccount {
    private double balance;

    public BankAccount(double initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(double amount) {
        balance += amount;
        System.out.println("જમા કર્યું: રૂ." + amount);
        System.out.println("વર્તમાન બેલેન્સ: રૂ." + balance);
    }

    public void withdraw(double amount) throws InsufficientFundException {
        if (amount > balance) {
            throw new InsufficientFundException("પૂરતું ભંડોળ નથી");
        }
        balance -= amount;
        System.out.println("ઉપાડ્યું: રૂ." + amount);
        System.out.println("બાકી બેલેન્સ: રૂ." + balance);
    }

    public double getBalance() { return balance; }
}

public class BankingApplication {
    public static void main(String[] args) {
        BankAccount account = new BankAccount(0);
        try {
            account.deposit(25000);
            account.withdraw(20000);
            account.withdraw(4000);
            account.withdraw(2000); // Throws exception
        } catch (InsufficientFundException e) {
            System.out.println("Exception: " + e.getMessage());
            System.out.println("ઉપલબ્ધ બેલેન્સ: રૂ." + account.getBalance());
        }
    }
}
```

મેમરી ટ્રીક

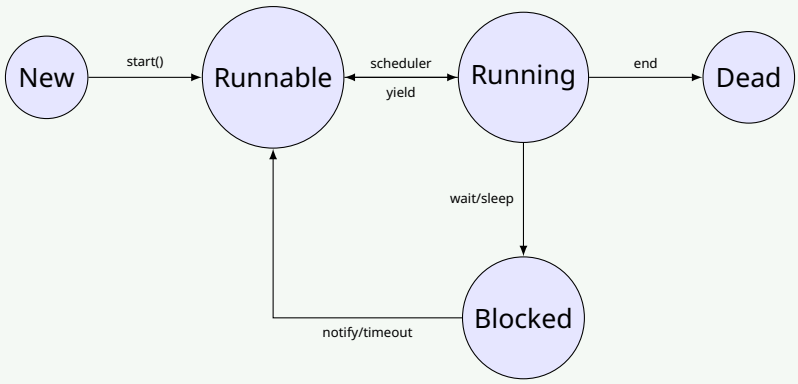
``Custom Throw Try-catch Message, Deposit Withdraw Check Handle''

પ્રશ્ન 4(અ OR) [3 ગુણ]

થ્રેડના સંપૂર્ણ જીવનચક્રનું વર્ણન કરો.

જવાબ

Thread Lifecycle States:



આકૃતિ 1. Thread Lifecycle

- **New**: Thread object બન્યો પણ start થયો નથી
- **Runnable**: Thread run થવા માટે તૈયાર
- **Running**: Thread હાલમાં execute થઈ રહ્યો છે
- **Blocked**: Thread રાહ જોઈ રહ્યો
- **Dead**: Thread execution પૂર્ણ થયું

મેમરી ટ્રીક

``New Runnable Running Blocked Dead''

પ્રશ્ન 4(બ OR) [4 ગુણ]

એક્સેસ સ્પેસીફાયરની યાદી બનાવો અને Java માં તેમના હેતુનું વર્ણન કરો.

જવાબ

કોષ્ટક 7. એક્સેસ સ્પેસિફાયર

એક્સેસ સ્પેસિફાયર	ક્લાસ	પેકેજ	સબક્લાસ	જગત
private	✓	×	×	×
default	✓	✓	×	×
protected	✓	✓	✓	×
public	✓	✓	✓	✓

મેમરી ટ્રીક

``Private Encapsulates, Default Packages, Protected inherits, Public Universal''

પ્રશ્ન 4(ક OR) [7 ગુણ]

Java પ્રોગ્રામ લખો જે 2 થ્રેડો ચલાવે છે દર થ્રેડ એક 1000 મીલીસેકન્ડે ``થ્રેડ --I'' ને પ્રિન્ટ કરે છે, બીજો થ્રેડ દર 2000 મીલીસેકન્ડે -- થ્રેડ ``II'' ને પ્રિન્ટ કરે છે. થ્રેડ Class ને એક્સટેન્ડ કરીને થ્રેડ બનાવો.

જવાબ

Listing 16. Multithreading Demo

```
class Thread1 extends Thread {
   public void run() {
      try {
         for (int i = 1; i <= 10; i++) {
            System.out.println("Thread1 - Count: " + i);
            Thread.sleep(1000); // 1000 ms
         }
      } catch (InterruptedException e) {
         System.out.println("Thread1 interrupted");
      }
   }
}

class Thread2 extends Thread {
   public void run() {
      try {
         for (int i = 1; i <= 5; i++) {
            System.out.println("Thread2 - Count: " + i);
            Thread.sleep(2000); // 2000 ms
         }
      } catch (InterruptedException e) {
         System.out.println("Thread2 interrupted");
      }
   }
}

public class MultiThreadDemo {
   public static void main(String[] args) {
      Thread1 t1 = new Thread1();
      Thread2 t2 = new Thread2();

      System.out.println("Starting threads...");
      t1.start();
      t2.start();

      try {
         t1.join();
         t2.join();
      } catch (InterruptedException e) {
         System.out.println("Main interrupted");
      }
      System.out.println("Done");
   }
}
```

મેમરી ટ્રીક

``Extend Run Sleep Start Join''

પ્રશ્ન 5(અ) [3 ગુણ]

સ્ટ્રીમ ક્લાસ શું છે? સ્ટ્રીમ ક્લાસ કેવી રીતે વર્ગીકૃત કરવામાં આવે છે?

જવાબ

Stream Class વ્યાખ્યા: Java માં Stream classes input અને output operations handle કરે છે.
Stream Classification:
- **Direction**: Input vs Output
- **Data Type**: Byte vs Character

કોષ્ટક 8. Stream Types

Type	Purpose	Examples
InputStream	Bytes વાંચવા	FileInputStream
OutputStream	Bytes લખવા	FileOutputStream
Reader	Characters વાંચવા	FileReader
Writer	Characters લખવા	FileWriter

મેમરી ટ્રીક

``Direction Data-type Functionality''

પ્રશ્ન 5(બ) [4 ગુણ]

ઉદાહરણ સાથે મેથડ ઓવરરાઇડિંગનો હેતુ સમજાવો.

જવાબ

Method Overriding નો હેતુ: Subclass ને specific implementation આપવા માટે.

Listing 17. Overriding Purpose

```
class Shape {
    public void draw() { System.out.println("Drawing generic shape"); }
}

class Circle extends Shape {
    @Override
    public void draw() { System.out.println("Drawing circle"); }
}

class Rectangle extends Shape {
    @Override
    public void draw() { System.out.println("Drawing rectangle"); }
}

public class OverridingDemo {
    public static void main(String[] args) {
        Shape s1 = new Circle();
        Shape s2 = new Rectangle();
```

```
        s1.draw(); // Calls Circle's draw
        s2.draw(); // Calls Rectangle's draw
    }
}
```

મેમરી ટ્રીક

``Runtime Specific Flexibility Dynamic''

પ્રશ્ન 5(ક) [7 ગુણ]

Abc.txt નામની ટેક્સ્ટ ફાઇલ પર વાંચવા અને લખવાની કામગીરી કરવા માટેનો Java પ્રોગ્રામ લખો.

જવાબ

Listing 18. File Operations

```
import java.io.*;

public class FileOperations {
    public static void main(String[] args) {
        String fileName = "Abc.txt";

        try {
            // Write
            FileWriter writer = new FileWriter(fileName);
            writer.write("Hello, this is Java file handling.\n");
            writer.write("End of file content.");
            writer.close();
            System.out.println("Data written successfully.");

            // Read
            FileReader reader = new FileReader(fileName);
            BufferedReader bufferedReader = new BufferedReader(reader);
            System.out.println("Reading from file:");
            String line;
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println(line);
            }
            bufferedReader.close();
        } catch (IOException e) {
            System.out.println("IO Error: " + e.getMessage());
        }
    }
}
```

મેમરી ટ્રીક

``Create Perform Handle Close''

પ્રશ્ન 5(અ OR) [3 ગુણ]

ઇનપુટ સ્ટ્રીમ સમજાવો.

જવાબ

InputStream: Abstract class જે bytes ના input stream ને represent કરે છે.
Methods:
- read(): Reads single byte
- read(byte[]): Reads bytes into array
- close(): Closes the stream
- available(): Returns estimated available bytes

મેમરી ટ્રીક

``Abstract Byte Input Resource''

પ્રશ્ન 5(બ OR) [4 ગુણ]

Java માં પેકેજ વ્યાખ્યાયિત કરો. યોગ્ય સીન્ટેક્સ અને એક ઉદાહરણ સાથે Java માં પેકેજ કેવી રીતે ઇમ્પ્લીમેન્ટ કરી શકાય તે લખો.

જવાબ

Package: Namespace જે classes organize કરે છે.
Steps:
- **Declare**: package name;
- **Directory**: Create folder matching package name
- **Compile**: Use javac -d . File.java

મેમરી ટ્રીક

``Namespace Access Organization Reusability''

પ્રશ્ન 5(ક OR) [7 ગુણ]

List નો ઉપયોગ દર્શાવવા માટે Java મા પ્રોગ્રામ લખો. 1) ArrayList બનાવો અને અઠવાડિયાના દિવસો ઉમેરો (સ્ટ્રિંગ સ્વરૂપમા) 2) LinkedList બનાવો અને મહિના ઉમેરો (સ્ટ્રિંગ સ્વરૂપમા) બંને List ને ડિસ્પ્લે કરો.

જવાબ

Listing 19. List Demo

```
import java.util.*;

public class ListDemo {
   public static void main(String[] args) {
      // 1. ArrayList for Weekdays
      ArrayList<String> weekdays = new ArrayList<>();
      weekdays.add("Monday");
      weekdays.add("Tuesday");
      weekdays.add("Wednesday");

      System.out.println("Weekdays (ArrayList):");
      for(String day : weekdays) System.out.println(day);

      // 2. LinkedList for Months
      LinkedList<String> months = new LinkedList<>();
      months.add("January");
```

```
        months.add("February");
        months.add("March");

        System.out.println("\nMonths (LinkedList):");
        for(String month : months) System.out.println(month);

        // Extra operations
        System.out.println("\nFirst Month: " + months.getFirst());
        System.out.println("ArrayList Size: " + weekdays.size());
    }
}
```

કોષ્ટક 9. ArrayList vs LinkedList

Feature	ArrayList	LinkedList
Structure	Dynamic Array	Doubly Linked List
Access	Fast O(1)	Slow O(n)
Insert/Delete	Slow (Shift needed)	Fast (Ptr change)

મેમરી ટ્રીક

``Dynamic Ordered Duplicate Index-based''