BSC. (HONS) IN ELECTRONICS AND TELECOMMUNICATIONS ENGINEERING

ECS2301 – Software Engineering and Project

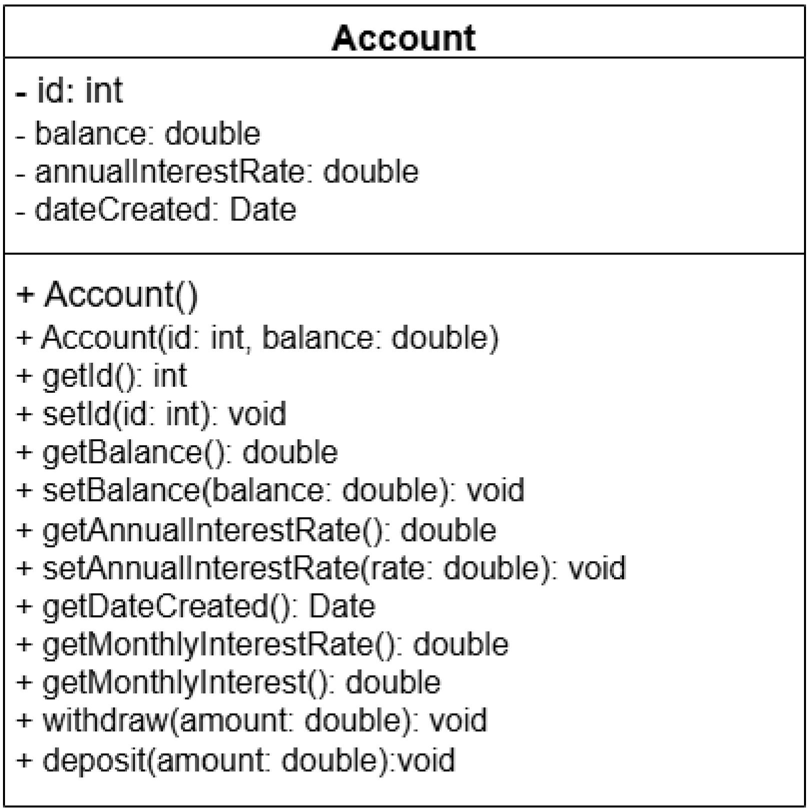
LAB ASSIGNMENT NO. : 03

INDEX NUMBER : 23UG1- 0152_Akindu Randira

17th JANUARY 2024

Q_01)

- **UML Diagram**



Account
- id: int - balance: double - annualInterestRate: double - dateCreated: Date
+ Account() + Account(id: int, balance: double) + getId(): int + setId(id: int): void + getBalance(): double + setBalance(balance: double): void + getAnnualInterestRate(): double + setAnnualInterestRate(rate: double): void + getDateCreated(): Date + getMonthlyInterestRate(): double + getMonthlyInterest(): double + withdraw(amount: double): void + deposit(amount: double):void

- **Implementation code of the Account Class in Java**

```
import java.util.Date;


public class Account {
  // Private fields
  private int id = 0;
  private double balance = 0;
  private double annualInterestRate = 0; // Annual interest rate in percentage
  private Date dateCreated;


  // No-arg constructor
  public Account() {
    this.dateCreated = new Date();
  }
```

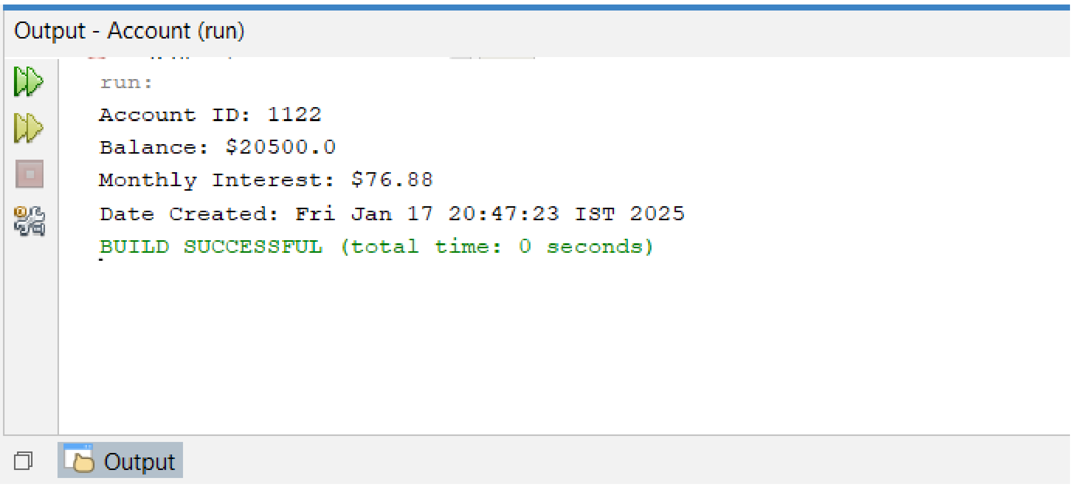
```
// Constructor with specified id and balance

public Account(int id, double balance) {

   this.id = id;

   this.balance = balance;

   this.dateCreated = new Date();

}


// Accessor and mutator methods for id

public int getId() {

   return id;

}


public void setId(int id) {

   this.id = id;

}


// Accessor and mutator methods for balance

public double getBalance() {

   return balance;

}


public void setBalance(double balance) {

   this.balance = balance;

}


// Accessor and mutator methods for annualInterestRate

public double getAnnualInterestRate() {

   return annualInterestRate;

}
```

```
    public void setAnnualInterestRate(double annualInterestRate) {
        this.annualInterestRate = annualInterestRate;
    }

    // Accessor method for dateCreated
    public Date getDateCreated() {
        return dateCreated;
    }

    // Method to get the monthly interest rate
    public double getMonthlyInterestRate() {
        return annualInterestRate / 12 / 100;
    }

    // Method to get the monthly interest
    public double getMonthlyInterest() {
        return balance * getMonthlyInterestRate();
    }
    // Method to withdraw a specified amount
    public void withdraw(double amount) {
        if (amount > balance) {
            System.out.println("Insufficient balance to withdraw $" + amount);
        } else {
            balance -= amount;
        }
    }
    // Method to deposit a specified amount
    public void deposit(double amount) {
        balance += amount;
    }
}
```

- ## Test code for Account

```
public class TestAccount {

  public static void main(String[] args) {

    // Create an Account object

    Account account = new Account(1122, 20000);

    account.setAnnualInterestRate(4.5);



    // Perform a withdrawal of $2,500

    account.withdraw(2500);



    // Perform a deposit of $3,000

    account.deposit(3000);



    // Print the balance, monthly interest, and account creation date

    System.out.println("Account ID: " + account.getId());

    System.out.println("Balance: $" + account.getBalance());

    System.out.printf("Monthly Interest: $%.2f%n", account.getMonthlyInterest());

    System.out.println("Date Created: " + account.getDateCreated());

  }

}
```

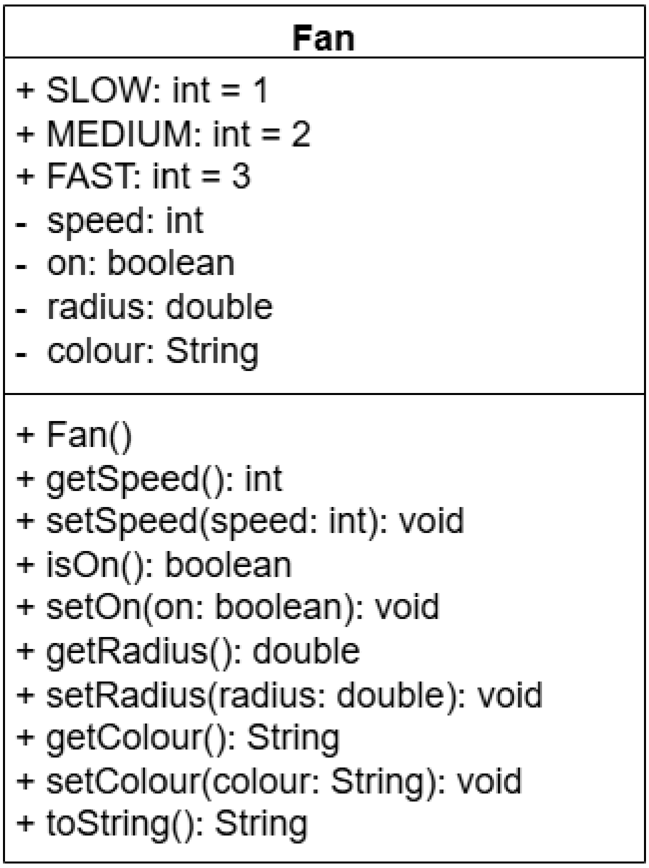
- ## Output



```
run:
Account ID: 1122
Balance: $20500.0
Monthly Interest: $76.88
Date Created: Fri Jan 17 20:47:23 IST 2025
BUILD SUCCESSFUL (total time: 0 seconds)
```

Q_02)

- **UML Diagram**



Fan

+ SLOW: int = 1
+ MEDIUM: int = 2
+ FAST: int = 3
- speed: int
- on: boolean
- radius: double
- colour: String

+ Fan()
+ getSpeed(): int
+ setSpeed(speed: int): void
+ isOn(): boolean
+ setOn(on: boolean): void
+ getRadius(): double
+ setRadius(radius: double): void
+ getColour(): String
+ setColour(colour: String): void
+ toString(): String

- **Implementation code of the Fan Class in Java**

```
public class Fan {

  // Constants for fan speeds

  public static final int SLOW = 1;

  public static final int MEDIUM = 2;

  public static final int FAST = 3;



  // Private fields

  private int speed = SLOW;

  private boolean on = false;

  private double radius = 5.0;

  private String color = "blue";
```

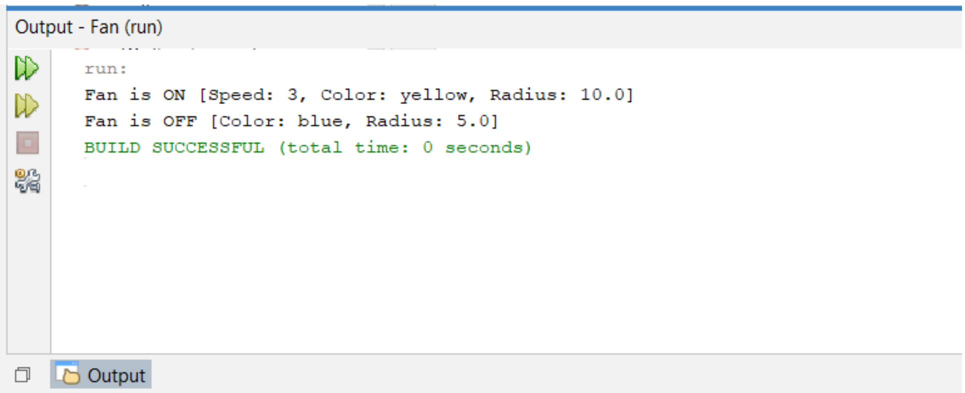
```
// No-arg constructor
public Fan() {
}

// Accessor and mutator for speed
public int getSpeed() {
   return speed;
}

public void setSpeed(int speed) {
   this.speed = speed;
}

// Accessor and mutator for on
public boolean isOn() {
   return on;
}

public void setOn(boolean on) {
   this.on = on;
}

// Accessor and mutator for radius
public double getRadius() {
   return radius;
}

public void setRadius(double radius) {
   this.radius = radius;
}
```

```
    // Accessor and mutator for color
    public String getColor() {
        return color;
    }

    public void setColor(String color) {
        this.color = color;
    }

    // toString method
    @Override
    public String toString() {
        if (on) {
            return "Fan is ON [Speed: " + speed + ", Color: " + color + ", Radius: " + radius + "]";
        } else {
            return "Fan is OFF [Color: " + color + ", Radius: " + radius + "]";
        }
    }
}
```

- **Test code for Fan**

```
public class TestFan {
    public static void main(String[] args) {
        // Create two Fan objects
        Fan fan1 = new Fan();
        Fan fan2 = new Fan();

        // Assign maximum speed, radius 10, color yellow, and turn it on for fan1
        fan1.setSpeed(Fan.FAST);
        fan1.setRadius(10.0);
        fan1.setColor("yellow");
        fan1.setOn(true);


        // Assign medium speed, radius 5, color blue, and turn it off for fan2

        fan2.setSpeed(Fan.MEDIUM);

        fan2.setRadius(5.0);

        fan2.setColor("blue");

        fan2.setOn(false);


        // Display the objects

        System.out.println(fan1);

        System.out.println(fan2);

    }

}
```

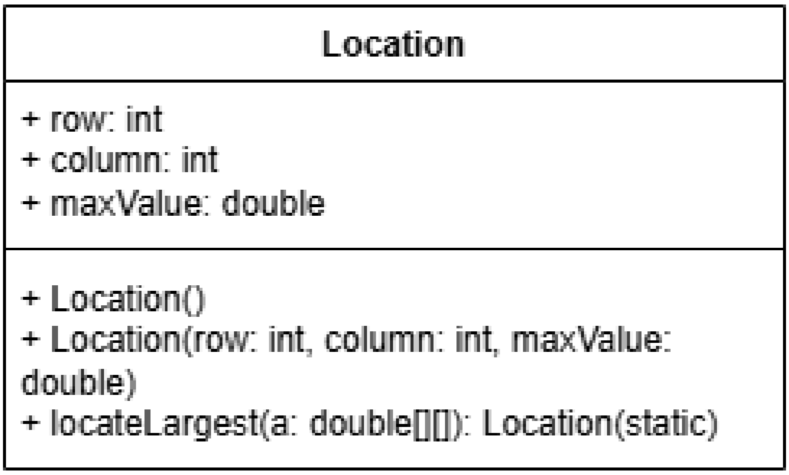
- **Output**



```
run:
Fan is ON [Speed: 3, Color: yellow, Radius: 10.0]
Fan is OFF [Color: blue, Radius: 5.0]
BUILD SUCCESSFUL (total time: 0 seconds)
```

Q_03)

- **UML Diagram**



Location

+ row: int
+ column: int
+ maxValue: double

+ Location()
+ Location(row: int, column: int, maxValue: double)
+ locateLargest(a: double[][]): Location(static)

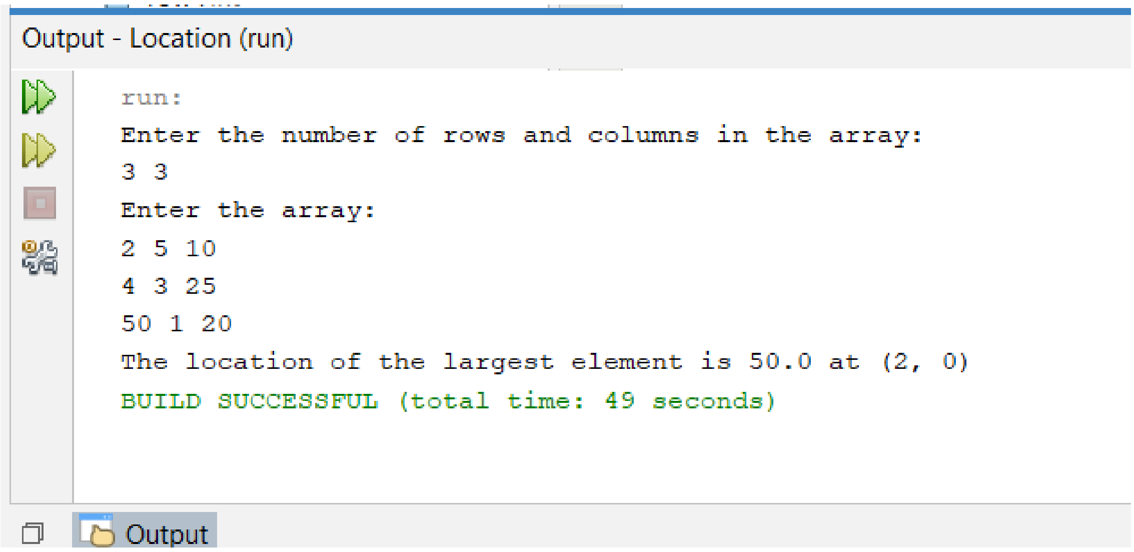
- **Implementation code of Location Class**

```
public class Location {
  public int row;
  public int column;
  public double maxValue;


  // Constructor to initialize the location
  public Location(int row, int column, double maxValue) {
    this.row = row;
    this.column = column;
    this.maxValue = maxValue;
  }


  // Static method to locate the largest element in a 2D array
  public static Location locateLargest(double[][] a) {
    int maxRow = 0;
    int maxColumn = 0;
    double max = a[0][0];


    for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[i].length; j++) {
                if (a[i][j] > max) {
                    max = a[i][j];
                    maxRow = i;
                    maxColumn = j;
                }
            }
        }
        return new Location(maxRow, maxColumn, max);
    }
}
```

- ## Test Program for Location

```
import java.util.Scanner;


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


        // Prompt user to enter the number of rows and columns
        System.out.println("Enter the number of rows and columns in the array:");
        int rows = input.nextInt();
        int columns = input.nextInt();
        // Create and fill the array
        double[][] array = new double[rows][columns];
        System.out.println("Enter the array:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                array[i][j] = input.nextDouble();
            }
        }
        // Locate the largest element
        Location location = Location.locateLargest(array);


        // Display the result
        System.out.printf("The location of the largest element is %.1f at (%d, %d)%n",
            location.maxValue, location.row, location.column);
    }
}
```

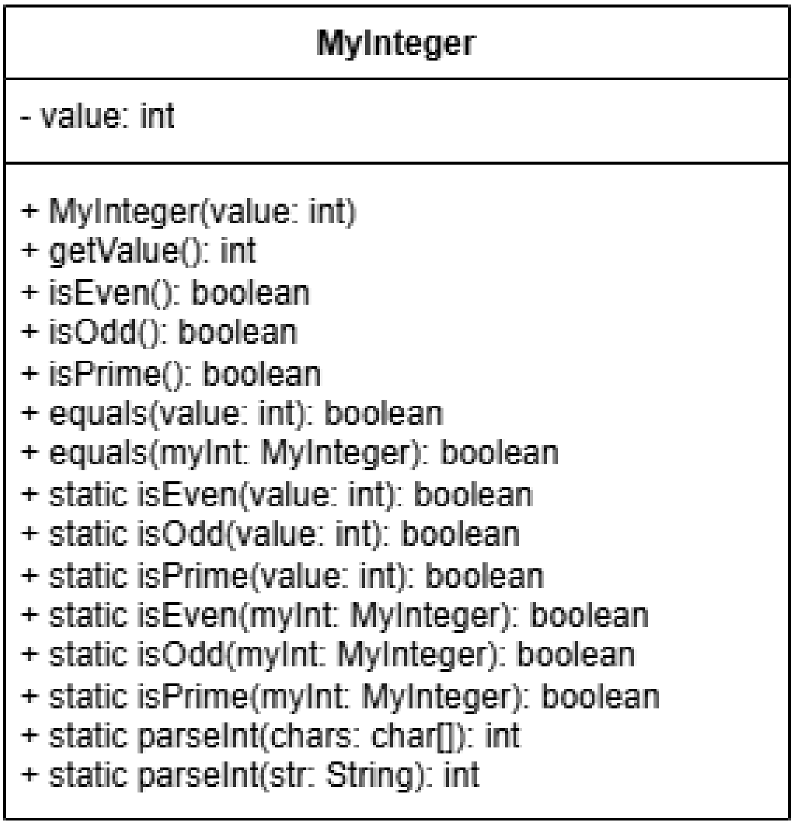
- ## Output



```
run:
Enter the number of rows and columns in the array:
3 3
Enter the array:
2 5 10
4 3 25
50 1 20
The location of the largest element is 50.0 at (2, 0)
BUILD SUCCESSFUL (total time: 49 seconds)
```

Q_04)

- **UML Diagram**



MyInteger

- value: int

+ MyInteger(value: int)
+ getValue(): int
+ isEven(): boolean
+ isOdd(): boolean
+ isPrime(): boolean
+ equals(value: int): boolean
+ equals(myInt: MyInteger): boolean
+ static isEven(value: int): boolean
+ static isOdd(value: int): boolean
+ static isPrime(value: int): boolean
+ static isEven(myInt: MyInteger): boolean
+ static isOdd(myInt: MyInteger): boolean
+ static isPrime(myInt: MyInteger): boolean
+ static parseInt(chars: char[]): int
+ static parseInt(str: String): int

- **Implementation code of MyInteger Class**

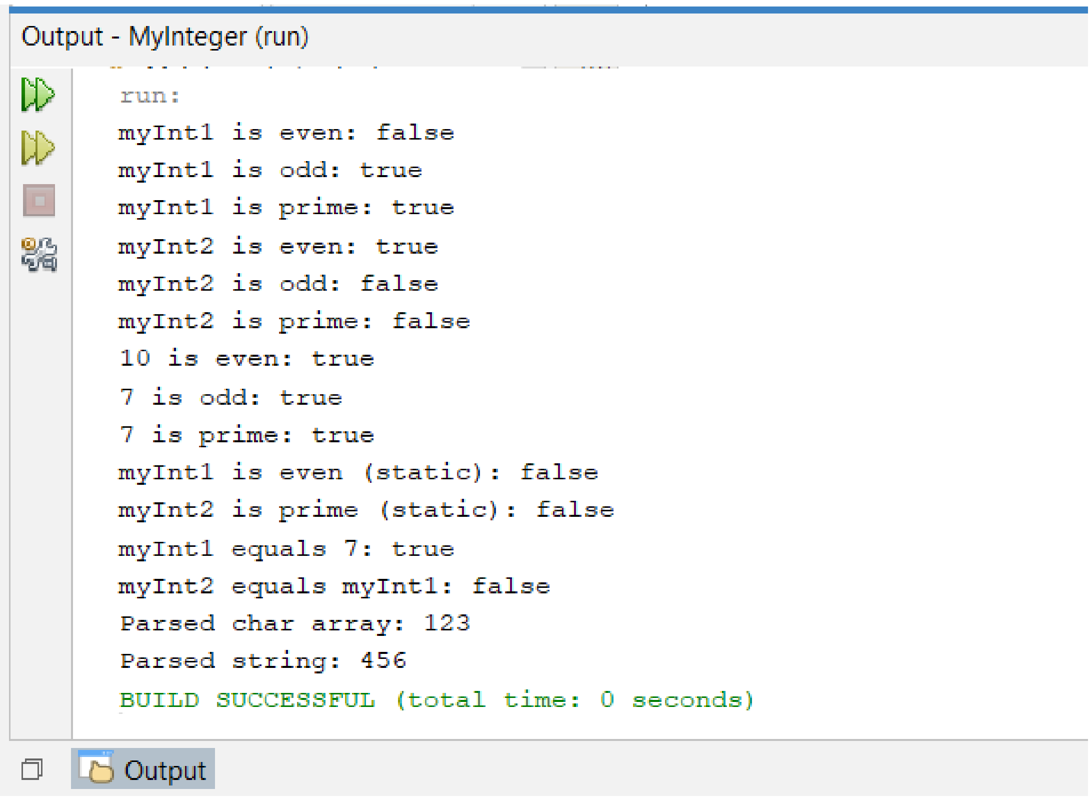
```
public class MyInteger {

  // Data field to store the integer value

  private int value;



  // Constructor

  public MyInteger(int value) {

    this.value = value;

  }



  // Getter for value

  public int getValue() {

    return value;

  }
```

```
// Instance methods to check if the value is even, odd, or prime

public boolean isEven() {

    return value % 2 == 0;

}


public boolean isOdd() {

    return value % 2 != 0;

}


public boolean isPrime() {

    if (value < 2) return false;

    for (int i = 2; i <= Math.sqrt(value); i++) {

        if (value % i == 0) return false;

    }

    return true;

}


// Static methods for int values

public static boolean isEven(int value) {

    return value % 2 == 0;

}


public static boolean isOdd(int value) {

    return value % 2 != 0;

}


public static boolean isPrime(int value) {

    if (value < 2) return false;

    for (int i = 2; i <= Math.sqrt(value); i++) {

        if (value % i == 0) return false;
    }
    return true;
}

// Static methods for MyInteger objects
public static boolean isEven(MyInteger myInt) {
    return myInt.isEven();
}

public static boolean isOdd(MyInteger myInt) {
    return myInt.isOdd();
}

public static boolean isPrime(MyInteger myInt) {
    return myInt.isPrime();
}

// Equals methods
public boolean equals(int value) {
    return this.value == value;
}

public boolean equals(MyInteger myInt) {
    return this.value == myInt.value;
}

// Static method to parse char array to int
public static int parseInt(char[] chars) {
    int result = 0;
    for (char c : chars) {
        result = result * 10 + (c - '0');
    }
    return result;
}

// Static method to parse String to int
public static int parseInt(String str) {
    return Integer.parseInt(str);
}
}
```

- ## Test Program for MyInteger

```
public class TestMyInteger {
    public static void main(String[] args) {
        // Create MyInteger objects
        MyInteger myInt1 = new MyInteger(7);
        MyInteger myInt2 = new MyInteger(10);

        // Test instance methods
        System.out.println("myInt1 is even: " + myInt1.isEven());
        System.out.println("myInt1 is odd: " + myInt1.isOdd());
        System.out.println("myInt1 is prime: " + myInt1.isPrime());

        System.out.println("myInt2 is even: " + myInt2.isEven());
        System.out.println("myInt2 is odd: " + myInt2.isOdd());
        System.out.println("myInt2 is prime: " + myInt2.isPrime());

        // Test static methods for int
        System.out.println("10 is even: " + MyInteger.isEven(10));
        System.out.println("7 is odd: " + MyInteger.isOdd(7));
        System.out.println("7 is prime: " + MyInteger.isPrime(7));
        // Test static methods for MyInteger

        System.out.println("myInt1 is even (static): " + MyInteger.isEven(myInt1));

        System.out.println("myInt2 is prime (static): " + MyInteger.isPrime(myInt2));



        // Test equals methods

        System.out.println("myInt1 equals 7: " + myInt1.equals(7));

        System.out.println("myInt2 equals myInt1: " + myInt2.equals(myInt1));



        // Test parseInt methods

        char[] chars = {'1', '2', '3'};

        System.out.println("Parsed char array: " + MyInteger.parseInt(chars));



        String str = "456";

        System.out.println("Parsed string: " + MyInteger.parseInt(str));

    }

}
```

- **Output**



```
run:
myInt1 is even: false
myInt1 is odd: true
myInt1 is prime: true
myInt2 is even: true
myInt2 is odd: false
myInt2 is prime: false
10 is even: true
7 is odd: true
7 is prime: true
myInt1 is even (static): false
myInt2 is prime (static): false
myInt1 equals 7: true
myInt2 equals myInt1: false
Parsed char array: 123
Parsed string: 456
BUILD SUCCESSFUL (total time: 0 seconds)
```