Here are the solutions to the exam questions.

1. Score to Grade Conversion Program

This program converts a numeric score into a letter grade according to the provided table using a switch...case statement. To handle the score ranges, the score is divided by 10.

Java

```java
import java.util.Scanner;

public class ScoreConverter {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Masukkan skor (0-100): ");
        int skor = input.nextInt();
        String nilai;

        // To use switch for ranges, we can group the scores by dividing by 10.
        int adjustedScore = skor / 10;

        switch (adjustedScore) {
            case 10:
            case 9:
            case 8:
                nilai = "A";
                break;
            case 7:
                nilai = "B";
                break;
            case 6:
                nilai = "C";
                break;
            case 5:
            case 4:
                nilai = "D";
                break;
```

```
            default:
                // This covers scores from 0 to 39
                nilai = "E";
                break;
        }

        System.out.println("Skor: " + skor + ", Nilai: " + nilai);
        input.close();
    }
}
```

---

2. Array Operations

This program demonstrates the creation, display (using for and while loops), and addition of two-dimensional arrays. A 3x3 array is used as an example.

Java

```
public class ArrayOperations {
    public static void main(String[] args) {
        [cite_start]// Create two 3x3 arrays [cite: 45, 46]
        int[][] data1 = {
            {5, 7, 2},
            {9, 3, 4},
            {8, 1, 6}
        };

        int[][] data2 = {
            {1, 2, 3},
            {4, 5, 6},
            {7, 8, 9}
        };

        // a. [cite_start]Display data1 using for loop [cite: 47, 48]
        System.out.println("Menampilkan data1 dengan for loop:");
        for (int i = 0; i < data1.length; i++) {
```

```
            for (int j = 0; j < data1[i].length; j++) {
                System.out.print(data1[i][j] + "\t");
            }
            System.out.println();
        }

        // b. [cite_start]Display data1 using while loop [cite: 47, 49]
        System.out.println("\nMenampilkan data1 dengan while loop:");
        int i = 0;
        while (i < data1.length) {
            int j = 0;
            while (j < data1[i].length) {
                System.out.print(data1[i][j] + "\t");
                j++;
            }
            System.out.println();
            i++;
        }

        // c. [cite_start]Add data1 and data2 [cite: 50]
        int[][] hasil = new int[3][3];
        for (int row = 0; row < data1.length; row++) {
            for (int col = 0; col < data1[row].length; col++) {
                hasil[row][col] = data1[row][col] + data2[row][col];
            }
        }

        // d. [cite_start]Display the result using for loop [cite: 51]
        System.out.println("\nHasil Penjumlahan data1 + data2:");
        for (int row = 0; row < hasil.length; row++) {
            for (int col = 0; col < hasil[row].length; col++) {
                System.out.print(hasil[row][col] + "\t");
            }
            System.out.println();
        }
    }
}
```

---

3. Object-Oriented Programming Implementation

This solution implements the class structure involving inheritance, method overriding, and method overloading as specified.

Note: The question contains some non-standard terminology. Specifically, a method that overrides another *must* have the same name. The prompt asks to create a method m3 that overrides m1 and then call m3. This is not possible in Java. The following code interprets the question's intent: it correctly overrides method m1 in class k3 and then calls this overridden version from a k3 object. It also creates a new method m4 in class k3 as requested.

Class k1

This class contains the base methods m1 and m2.

Java

```
// Kelas k1
public class k1 {
    // a. [cite_start]Metode m1 dengan return value dan access specifier public [cite: 54]
    public int m1(int a, int b) {
        System.out.println("Menjalankan m1 dari kelas k1...");
        return (3 * a) + (4 * b);
    }

    // b. [cite_start]Metode m2 tanpa return value dan access specifier protected [cite: 55, 56]
    protected void m2(int a, int b) {
        System.out.println("Menjalankan m2 dari kelas k1...");
        int hasil_m1 = m1(a, b); // Memanggil m1
        System.out.println("Hasil dari 2 * luaran m1 adalah: " + (2 * hasil_m1));
    }
}
```

Class k2

This class inherits from k1.

Java

```
// c. [cite_start]Kelas k2, turunan dari k1 [cite: 57]
public class k2 extends k1 {
    // Kelas ini mewarisi metode m1 dan m2 dari k1
}
```

Class k3

This class inherits from k2 and demonstrates overriding and adding a new method.

Java

```
// d. [cite_start]Kelas k3, turunan dari k2 [cite: 58]
public class k3 extends k2 {
    [cite_start]// Meng-override method m1 dari kelas k1 [cite: 58]
    // (Dalam soal disebut sebagai m3, namun overriding harus memiliki nama method yang sama)
    @Override
    public int m1(int a, int b) {
        System.out.println("Menjalankan m1 yang di-override dari kelas k3...");
        // Formula diubah untuk menunjukkan bahwa ini adalah metode yang berbeda
        return (5 * a) + (6 * b);
    }

    [cite_start]// Method m4 yang merupakan bentuk overloading dari m1 [cite: 59]
    // (Nama berbeda tetapi konsepnya adalah memiliki parameter berbeda)
    public int m4(int a, int b, int c) {
        System.out.println("Menjalankan m4 dari kelas k3...");
        return a + b + c;
    }
```

}

Class app

This is the main class to run and test the methods from the other classes.

Java

```
// Kelas utama untuk menjalankan program
public class app {
    public static void main(String[] args) {
        // e. [cite_start]Menginstansiasi kelas k2 dan memanggil m1 & m2 [cite: 60]
        System.out.println("--- Menggunakan Objek dari Kelas k2 ---");
        k2 objek_k2 = new k2();
        int hasil_m1_k2 = objek_k2.m1(10, 5); // Memanggil m1 dari k1
        System.out.println("Return value dari m1 di k2: " + hasil_m1_k2);
        objek_k2.m2(10, 5); // Memanggil m2 dari k1
        System.out.println();

        // f. [cite_start]Menjalankan method m3 (overridden m1) dan m4 [cite: 61]
        System.out.println("--- Menggunakan Objek dari Kelas k3 ---");
        k3 objek_k3 = new k3();
        // Memanggil m1 yang sudah di-override di k3 (menggantikan m3 pada soal)
        int hasil_m1_k3 = objek_k3.m1(10, 5);
        System.out.println("Return value dari m1 (overridden) di k3: " + hasil_m1_k3);

        // Memanggil m4 dari k3
        int hasil_m4_k3 = objek_k3.m4(10, 5, 2);
        System.out.println("Return value dari m4 di k3: " + hasil_m4_k3);
    }
}
```