JAVA

1) ERROR :

```
public class ArrayManipulation {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        for (int i = 0; i <= numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

ArrayIndexOutOfBounds Exception

CORRECTION :

```
public class ArrayManipulation {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        for (int i = 0; i < numbers.length; i++) {
            System.out.println(numbers[i]);
        }
    }
}
```

EXPLANATION:

- The `numbers` array has 5 elements (index 0 to 4).
- The loop iterates from 0 to numbers.length (5).
- At i = 5, the code tries to access `numbers[5]`, which doesn't exist, causing the exception.

---

2) ERROR :

```
class Car {
    private String make;
    private String model;

    public Car(String make, String model) {
        this.make = make;
        this.model = model;
    }
```

```
    public void start() {
        System.out.println("Starting the car.");
    }
}

public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Camry");
        car.start();
        car.stop();
    }
}
```

compilation error

CORRECTION :

```
class Car {
    private String make;
    private String model;
    public Car(String make, String model) {
        this.make = make;
        this.model = model;
    }
    public void start() {
        System.out.println("Starting the car.");
    }
}
public class Main {
    public static void main(String[] args) {
        Car car = new Car("Toyota", "Camry");
        car.start();
    }
}
```

EXPLANATION:

The error occurs because the code attempts to call a method named `stop()` on the `car` object, but there is no such method defined in the `Car` class. This will result in a compiler error stating that the method `stop()` is not found.

---

3)ERROR :

```
public class ExceptionHandling {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};

        try {
```

```
            System.out.println(numbers[10]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds.");
        }

        int result = divide(10, 0);
        System.out.println("Result: " + result);
    }

    public static int divide(int a, int b) {
        return a / b;
    }
}
```

Arthimetic exception

CORRECTION :

```
public class ExceptionHandling {
    public static void main(String[] args) {
        int[] numbers = {1, 2, 3, 4, 5};
        try {
            System.out.println(numbers[1]);
        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Array index out of bounds.");
        }
        try {
            int result = divide(10, 0);
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Division by zero is not allowed.");
        }
    }
    public static int divide(int a, int b) {
        if (b == 0) {
            throw new ArithmeticException("Division by zero");
        }
        return a / b;
    }
}
```

EXPLANATION:

ArithmeticException:

- The `divide` method attempts to divide by 0, which is not allowed in standard arithmetic. This results in an `ArithmeticException`.

---

4)ERROR :

NO error in the given code

- The code does not have any errors as such . However, there is a bug in the `fibonacci` function that prevents it from calculating large Fibonacci numbers accurately.
- ***Bug***: The bug lies in the recursive calls within the `fibonacci` function. Although the function correctly calculates small Fibonacci numbers, the repeated recursion for larger values leads to stack overflow errors due to exceeding the memory limit.

CORRECTION :

-->Memoization: Instead of directly calling the `fibonacci` function recursively, store previously calculated values in a dictionary.

```
public class Fibonacci {
    private static HashMap<Integer, Integer> memo = new HashMap<>();
    public static int fibonacci(int n) {
        if (memo.containsKey(n)) {
            return memo.get(n);
        }
        if (n <= 1) {
            memo.put(n, n);
            return n;
        } else {
            int value = fibonacci(n - 1) + fibonacci(n - 2);
            memo.put(n, value);
            return value;
        }
    }
    public static void main(String[] args) {
        int n = 6;
        int result = fibonacci(n);
        System.out.println("The Fibonacci number at position " + n + " is: " + result);
    }
}
```

-->***Iterative approach***: Instead of recursion, use an iterative approach with a loop to calculate the Fibonacci numbers. This method is more efficient and avoids stack overflow issues.

```
public class Fibonacci {
    public static int fibonacci(int n) {
        if (n <= 1) {
            return n;
        }
        int a = 0;
```

```
        int b = 1;
        int c;
        for (int i = 2; i <= n; i++) {
            c = a + b;
            a = b;
            b = c;
        }
        return c;
    }
    public static void main(String[] args) {
        int n = 6;
        int result = fibonacci(n);
        System.out.println("The Fibonacci number at position " + n + " is: " + result);
    }
}
```

EXPLANATION:

- Each recursive call in the function creates a new frame on the stack. For large values of `n`, the number of frames becomes very large, eventually exceeding the available stack memory. This results in a stack overflow error and the program crashes.

---

5)ERROR :

NO error in the given code

- The provided code for finding prime numbers appears to be functional but could be improved in terms of efficiency.
- ***Bug***: While the code correctly identifies prime numbers, the nested loop approach used to check divisibility is unnecessarily slow for larger numbers. This is because it iterates through all potential divisors up to the number itself, which can be computationally expensive.

CORRECTION :

- To improve the code's efficiency, replace the inner loop with a condition that checks divisibility only up to the square root of the number:

```
import java.util.*;
public class PrimeNumbers {
    public static List<Integer> findPrimes(int n) {
        List<Integer> primes = new ArrayList<>();
        for (int i = 2; i <= n; i++) {
            boolean isPrime = true;
            for (int j = 2; j*j <= i; j++) {
                if (i % j == 0) {
                    isPrime = false;
```

```
                break;
            }
        }
```