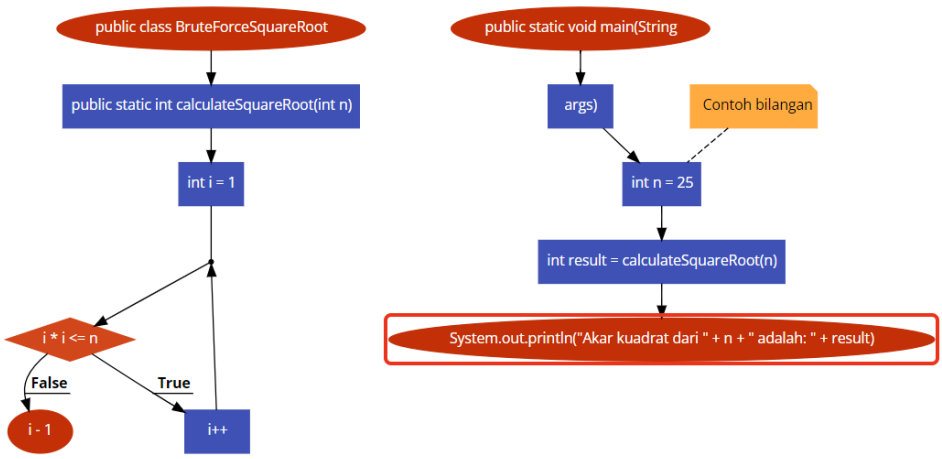
SAFRIZAL RAHMAN

Sib – 1G

19

Tugas p5

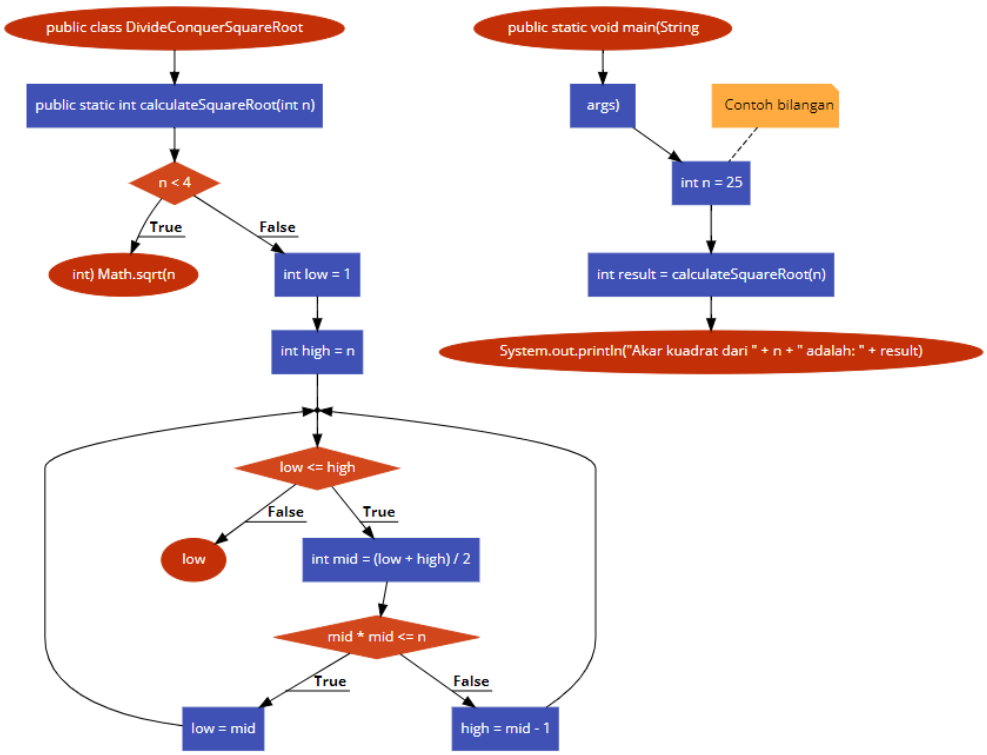
1. **A** Menghitung Akar Kuadrat (Brute Force)



start
input: bilangan (n)
inisialisasi: i = 1
loop:
  jika i^2 <= n:
    i += 1
  lainnya:
    keluar loop
akhir loop
output: i - 1 (akar kuadrat n)

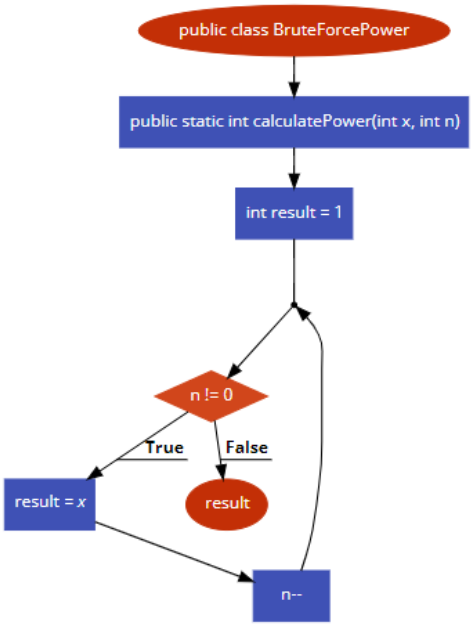
end

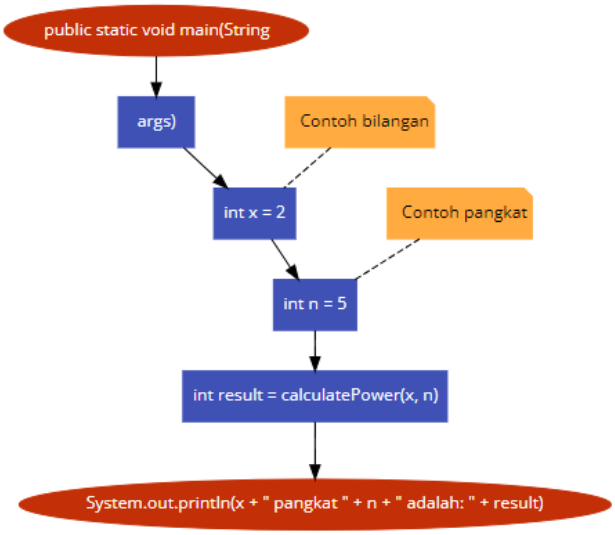
B Menghitung Akar Kuadrat (Divide Conquer)



start
input: bilangan (n)
jika n < 4:
  output: √n
lainnya:
  inisialisasi: low = 1, high = n
  loop:
    mid = (low + high) // 2
    jika mid^2 <= n:
      low = mid
    lainnya:
      high = mid - 1
  akhir loop
  output: low (akar kuadrat n)
end

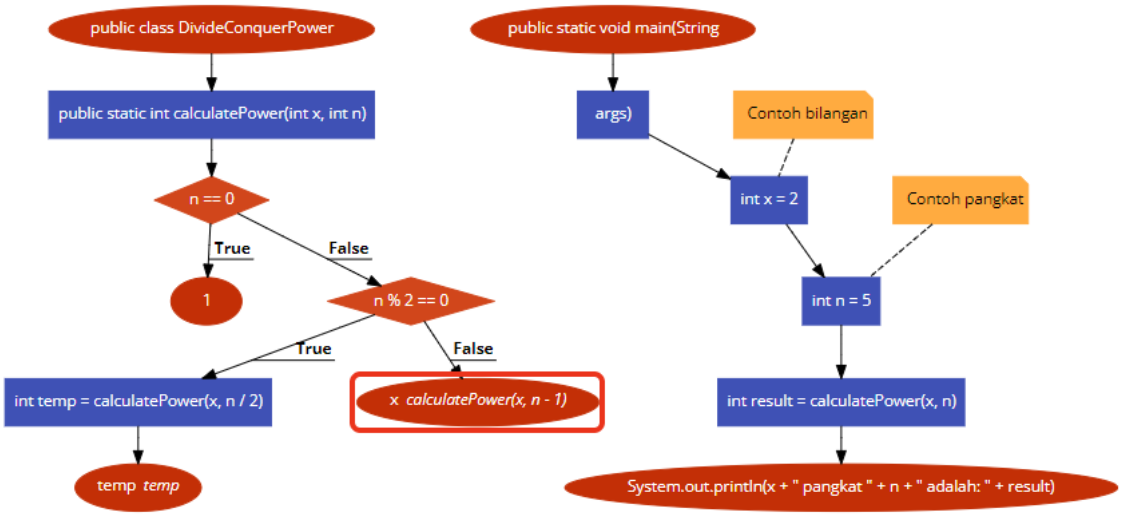
2. **A** Menghitung Pangkat (Brute Force)





start
input: bilangan (x), pangkat (n)
inisialisasi: hasil = 1
loop:
 jika n = 0:
  keluar loop
 lainnya:
  hasil = x
  n -= 1
 akhir loop
output: hasil (x pangkat n)
end

B Menghitung Pangkat (Divide Conquer)



start
input: bilangan (x), pangkat (n)
jika n = 0:
  output: 1
lainnya:
  jika n genap:
    hasil = (x^2)^(n/2)
  lainnya:
    hasil = x (x^(n-1))
  output: hasil (x pangkat n)
end

3. **A**

a.

```
public int countVowels(char[] word){
    char[] vowels = {'a', 'i', 'u', 'e', 'o'};
    int count = 0;

    for (int i = 0; i < word.length; i++) {
        for (int j = 0; j < vowels.length; j++) {
            if (word[i] == vowels[j]) {
                count++;
            }
        }
    }

    return count;
}
```

Analisis:
Terdapat dua loop bersarang dalam kode program ini.
Loop pertama (for (int i = 0; i < word.length; i++)) akan berjalan sebanyak panjang kata.
Loop kedua (for (int j = 0; j < vowels.length; j++)) akan berjalan sebanyak jumlah huruf vokal (yaitu 5). Konversi waktu program ini adalah O(n*m), di mana n adalah panjang kata dan m adalah jumlah huruf vokal. Notasi Big O: O(n*m)
di program ini memiliki kompleksitas waktu yang berkaitan dengan panjang kata dan jumlah huruf vokal. Semakin panjang kata dan semakin banyak huruf vokal, semakin lama program ini akan berjalan.

B

```java
public boolean checkItemInList(String item, String[] list) {
    for (int i = 0; i < list.length; i++) {
        if (list[i] == item) {
            return true;
        }
    }

    return false;
}
```

Inisialisasi: Metode ini menginisialisasi variabel integer `i` ke 0.
Perulangan atau foreach: Perulangan mengulang seluruh larik `daftar`, memeriksa setiap elemen terhadap `item`.
pembandingan: Di dalam perulangan, terdapat operasi perbandingan (`list[i] == item`). dan Jika kecocokan ditemukan, metode mengembalikan `true`; jika tidak, metode mengembalikan `false`.
Perulangan berjalan sepanjang panjang larik `daftar`, yang diwakili oleh `n` (di mana `n` adalah ukuran larik).
Operasi perbandingan (`list[i] == item`) membutuhkan waktu yang konstan, dilambangkan sebagai O(1).
Dimana waktu keseluruhan dari kode ini adalah O(n), di mana `n` merepresentasikan panjang larik `list`.