NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

LAPORAN PRAKTIKUM ALGORITMA DAN STRUKTUR DATA

2. 1. Percobaan 1: Penyimpanan Tumpukan Barang dalam Gudang

Class Barang :

PrakASD_1F_13 > src > P8 > Barang13.java > ...

```java
package P8;

Codeium: Refactor | Explain
public class Barang13 {
    int kode;
    String nama, kategori;

    public Barang13(int kode, String nama, String kategori) {
        this.kode = kode;
        this.nama = nama;
        this.kategori = kategori;
    }
}
```

Class Gudang :

PrakASD_1F_13 > src > P8 > Gudang13.java > Gudang13 > lihatBarangTeratas()

```java
package P8;

Codeium: Refactor | Explain
public class Gudang13 {
    Barang13[] tumpukan;
    int size;
    int top;

    public Gudang13(int kapasitas) {
        size = kapasitas;
        tumpukan = new Barang13[size];
        top = -1;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public boolean cekKosong() {
        if (top == -1) {
            return true;
        } else {
            return false;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public boolean cekPenuh() {
        if (top == size -1) {
            return true;
        } else {
            return false;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void tambahBarang(Barang13 brg) {
        if (!cekPenuh()) {
            top++;
            tumpukan[top] = brg;
            System.out.println("Barang " + brg.nama + " berhasil ditambahkan Ke Gudang");
        } else {
            System.out.println(x:"Gagal Tumpukan barang di Gudang sudah penuh");
        }
    }
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

```
    public Barang13 ambilBarang() {
        if (!cekKosong()) {
            Barang13 delete = tumpukan[top];
            top--;
            System.out.println("Barang " + delete.nama + " diambil dari Gudang.");
            return delete;
        } else {
            System.out.println(x:"Tumpukan barang kosong.");
            return null;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public Barang13 lihatBarangTeratas() {
        if (!isEmpty()) {
            Barang13 barangTeratas =  tumpukan[top];
            System.out.println("Barang teratas: " + barangTeratas.nama);
            return barangTeratas;
        } else {
            System.out.println(x:"Tumpukan barang kosong.");
            return null;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void tampilkanBarang() {
        if (!cekKosong()) {
            System.out.println(x:"Rincian tumpukan barang di Gudang: ");

            for (int i = 0; i <= top; i++) {
                System.out.printf(format:"Kode %d: %s (Kategori %s)\n", tumpukan[i].kode, tumpukan[i].nama,
                tumpukan[i].kategori);
            }
        } else {
            System.out.println(x:"Tumpukan barang kosong.");
        }
    }
}
```

Class Main :

```
PrakASD_1F_13 > src > P8 > Utama13.java > Utama13 > main(String[])
package P8;

import java.net.Socket;
import java.util.Scanner;;

Codeium: Refactor | Explain
public class Utama13 {
    Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
    public static void main(String[] args) {
        Gudang13 gudang = new Gudang13(kapasitas:7);

        Scanner input = new Scanner(System.in);

        while (true) {
            System.out.println(x:"\nMenu");
            System.out.println(x:"1. Tambah barang");
            System.out.println(x:"2. Ambil barang");
            System.out.println(x:"3. Tampil tumpukan barang");
            System.out.println(x:"4. Keluar");
            System.out.print(s:"Pilih operasi: ");
            int pilihan = input.nextInt();
            input.nextLine();

            switch (pilihan) {
                case 1:
                    System.out.print(s:"Masukkan kode barang: ");
                    int kode = input.nextInt();
                    input.nextLine();
                    System.out.print(s:"Masukkan nama barang: ");
                    String nama = input.nextLine();
                    System.out.print(s:"Masukkan nama kategori: ");
                    String kategori = input.nextLine();
                    Barang13 barangBaru = new Barang13(kode, nama, kategori);
                    gudang.tambahBarang(barangBaru);
                    break;
                case 2:
                    gudang.ambilBarang();
                    break;
                case 3:
                    gudang.tampilkanBarang();
                case 4:
                    break;
                default:
                    System.out.println(x:"Pilihan tidak valid. Silahkan coba lagi.");
            }
        }
    }
}
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

Output :

```
                                                Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 1
Masukkan kode barang: 21
Masukkan nama barang: Majalah
Masukkan nama kategori: Buku
Barang Majalah berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
4. Keluar
Pilih operasi: 1
Masukkan kode barang: 26
Masukkan nama kategori: Pakaian
Barang Jaket berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 2
Barang Jaket diambil dari Gudang.

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 1
Masukkan kode barang: 33
Masukkan nama barang: Pizza
Masukkan nama kategori: Makanan
Barang Pizza berhasil ditambahkan Ke Gudang
```

```
Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 3
Rincian tumpukan barang di Gudang:
Kode 21: Majalah (Kategori Buku)
Kode 33: Pizza (Kategori Makanan)

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi:
PS D:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

Masih belum sesuai.

```
Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 1
Masukkan kode barang: 21
Masukkan nama barang: Majalah
Masukkan nama kategori: Buku
Barang Majalah berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 1
Masukkan kode barang: 26
Masukkan nama barang: Jaket
Masukkan nama kategori: Pakaian
Barang Jaket berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 2
Barang Jaket diambil dari Gudang.
```

```
Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 1
Masukkan kode barang: 33
Masukkan nama barang: Pizza
Masukkan nama kategori: Makanan
Barang Pizza berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi: 3
Rincian tumpukan barang di Gudang:
Kode 33: Pizza (Kategori Makanan)
Kode 21: Majalah (Kategori Buku)

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Keluar
Pilih operasi:
```

sesuai

Question :

1. Lakukan perbaikan pada kode program, sehingga keluaran yang dihasilkan sama dengan verifikasi hasil percobaan! Bagian mana saja yang perlu diperbaiki?

   Jawab :

```
52      public Barang13 lihatBarangTeratas() {
53          if (!isEmpty) {
```

   Pada bagian itu digantikan dengan “cekKosong”

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

```
53          if (!cekKosong()) {
```

Dan bagian for tersebut karena saat menampilkan yang terakhir diinputkan berada di bawah.

```
67          for (int i = 0; i <= top; i++) {
68              System.out.printf(format:"Kode %d: %s (Kategori %s)\n", tumpukan[i].kode, tumpukan[i].nama,
```

Diganti sebagai berikut agar inputan yang terakhir berada di tumpukan atas.

```
67          for (int i = top; i >= 0; i--) {
```

Menambahkan “System.exit” agar case bisa digunakan.

```
                case 4:
                    System.exit(status:0);
```

2. Berapa banyak data barang yang dapat ditampung di dalam tumpukan? Tunjukkan potongan kode programnya!
   Jawab :

```
7           Gudang13 gudang = new Gudang13(kapasitas:7);
```

3. Mengapa perlu pengecekan kondisi !cekKosong() pada method tampilkanBarang? Kalau kondisi tersebut dihapus, apa dampaknya?
   Jawab : Untuk memastikan jika barang yang ditampilkan tidak kosong. Kalaupun kondisi dihilangkan akan terjadi error, dan jika tidak error(masih berjalan) maka tidak akan menampilakan hasil barangnya.

4. Modifikasi kode program pada class Utama sehingga pengguna juga dapat memilih operasi lihat barang teratas, serta dapat secara bebas menentukan kapasitas gudang!
   Jawab :

```
6       public static void main(String[] args) {
7           Scanner input = new Scanner(System.in);
8
9           System.out.print(s:"Masukkan jumlah kapasitas Gudang: ");
10          int jumKapasitas = input.nextInt();
11
12          Gudang13 gudang = new Gudang13(jumKapasitas);
13
14          while (true) {
15              System.out.println(x:"\nMenu");
16              System.out.println(x:"1. Tambah barang");
17              System.out.println(x:"2. Ambil barang");
18              System.out.println(x:"3. Tampil tumpukan barang");
19              System.out.println(x:"4. Melihat barang teratas");
20              System.out.println(x:"5. Keluar");
21              System.out.print(s:"Pilih operasi: ");
22              int pilihan = input.nextInt();
23              input.nextLine();
```

```
43              case 4:
44                  gudang.lihatBarangTeratas();
45                  break;
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

Output :

```
Masukkan jumlah kapasitas Gudang: 2

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Keluar
Pilih operasi: 1
Masukkan kode barang: 23
Masukkan nama barang: Aqua
Masukkan nama kategori: Minuman
Barang Aqua berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Keluar
Pilih operasi: 1
Masukkan kode barang: 24
Masukkan nama barang: Pizza
Masukkan nama kategori: Makanan
Barang Pizza berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Keluar
Pilih operasi: 4
Barang teratas: Pizza
```

5. Commit dan push kode program ke Github.
   Jawab : Sudah

2.2 Percobaan 2: Konversi Kode Barang ke Biner

Gudang

```
77      public String konversiDesimalKeBiner(int kode) {
78          StackKonversi13 stack = new StackKonversi13();
79          while (kode > 0) {
80              int sisa = kode % 2;
81              stack.push(sisa);
82              kode = kode / 2;
83          }
84          String biner = new String();
85          while (!stack.isEmty()) {
86              biner += stack.pop();
87          }
88          return biner;
89      }
90  }
91
```

NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: STACK

StackKonversi

PrakASD_1F_13 > src > P8 > StackKonversi13.java > StackKonversi13 > pop()

```java
package P8;

Codeium: Refactor | Explain
public class StackKonversi13 {
    int size;
    int[] tumpukanBiner;
    int top;

    public StackKonversi13() {
        this.size = 32;
        tumpukanBiner = new int[size];
        top = -1;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public boolean isEmty() {
        return top == -1;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public boolean isFull() {
        return top == size -1;
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public void push(int data) {
        if (isFull()) {
            System.out.println(x:"Stack penuh");
        } else {
            top++;
            tumpukanBiner[top] = data;
        }
    }

    Codeium: Refactor | Explain | Generate Javadoc | ×
    public int pop() {
        if (isEmty()) {
            System.out.println(x:"Stack kososng");
            return -1;
        } else {
            int data = tumpukanBiner[top];
            top--;
            return data;
        }
    }
}
```

Gudang(ambil Barang)

```java
    public Barang13 ambilBarang() {
        if (!cekKosong()) {
            Barang13 delete = tumpukan[top];
            top--;
            System.out.println("Barang " + delete.nama + " diambil dari Gudang.");
            System.out.println("Kode unik dalam biner: " + konversiDesimalKeBiner(delete.kode));
            return delete;
        } else {
            System.out.println(x:"Tumpukan barang kosong.");
            return null;
        }
    }
```

NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: STACK

Output:

```
Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Keluar
Pilih operasi: 1
Masukkan kode barang: 13
Masukkan nama barang: Setrika
Masukkan nama kategori: Elektronik
Barang Setrika berhasil ditambahkan Ke Gudang

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Keluar
Pilih operasi: 2
Barang Setrika diambil dari Gudang.
Kode unik dalam biner: 1101

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Keluar
Pilih operasi: 5
PS D:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

Question :

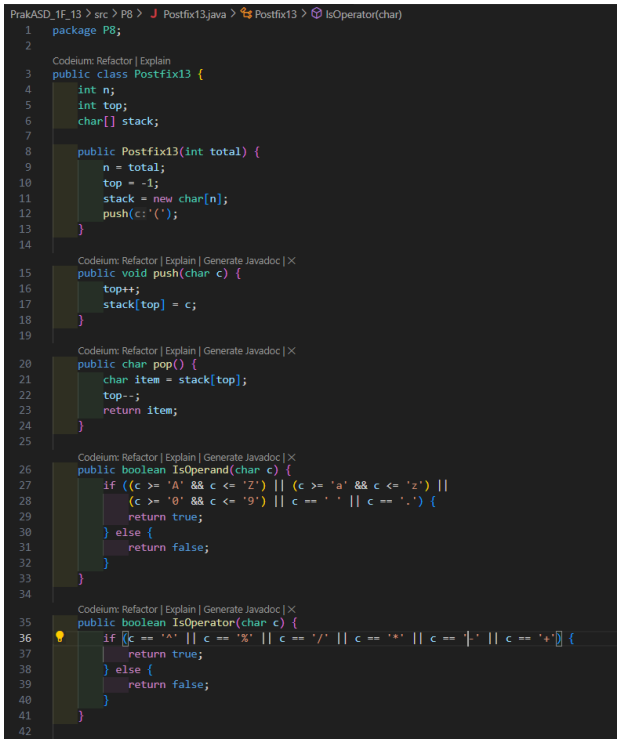
1. Pada method konversiDesimalKeBiner, ubah kondisi perulangan menjadi while (kode != 0), bagaimana hasilnya? Jelaskan alasannya!
   Jawab : Hasilnya sama saja, karena sama – sama lebih dari 0.

2. Jelaskan alur kerja dari method konversiDesimalKeBiner!
   Jawab :
   1. Akan melakukan kondisi (kode != 0) yang dimana kode tidak boleh 0.
   2. sisa untuk menampung hasil dari kode % 2;
   3. sisa tersebut menngepush di stack.
   4. kode dibagi 2 dan hasilnya akan dikonversi lagi.
   5. Setelah perulangan pertama selesai, lalu terjadi inialisasi biner.
   6. Melakukan perulangan kedua dengan kondisi (!isEmpty()).
   7. Elemen di stack dipindahkan secara terbalik menggunakan pop dan ditambahkan ke biner sampai stack kosong.

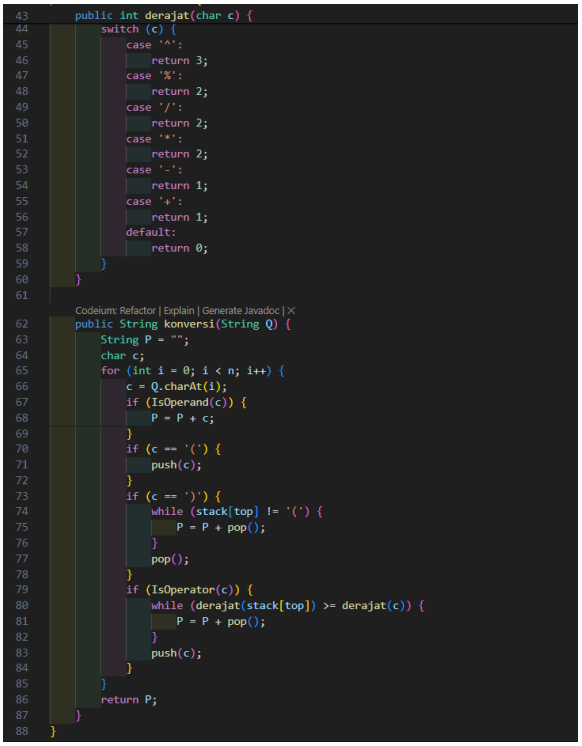
NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: STACK

2.3 Percobaan 3: Konversi Notasi Infix ke Postfix

Postfix

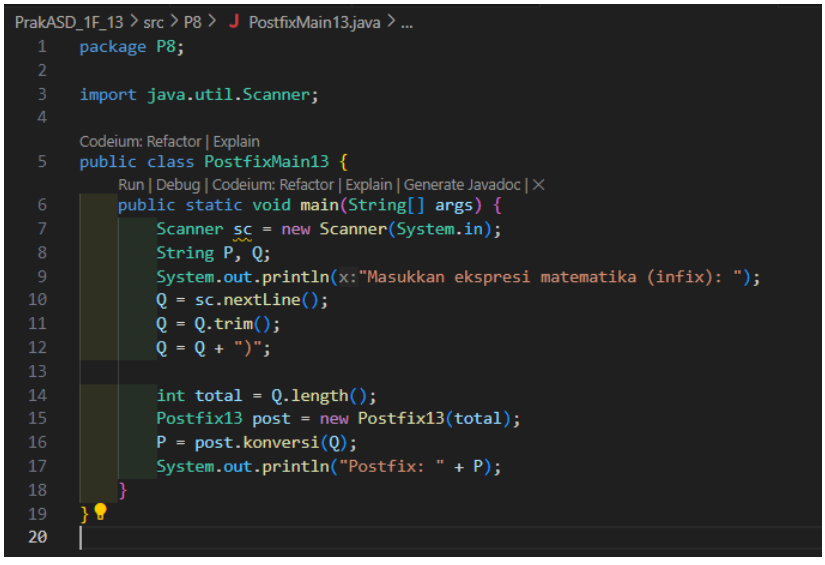


```
PrakASD_1F_13 > src > P8 > Postfix13.java > Postfix13 > IsOperator(char)
1   package P8;
2
    Codeium: Refactor | Explain
3   public class Postfix13 {
4       int n;
5       int top;
6       char[] stack;
7
8       public Postfix13(int total) {
9           n = total;
10          top = -1;
11          stack = new char[n];
12          push(c:'(');
13      }
14
    Codeium: Refactor | Explain | Generate Javadoc | ×
15      public void push(char c) {
16          top++;
17          stack[top] = c;
18      }
19
    Codeium: Refactor | Explain | Generate Javadoc | ×
20      public char pop() {
21          char item = stack[top];
22          top--;
23          return item;
24      }
25
    Codeium: Refactor | Explain | Generate Javadoc | ×
26      public boolean IsOperand(char c) {
27          if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z') ||
28              (c >= '0' && c <= '9') || c == ' ' || c == '.') {
29              return true;
30          } else {
31              return false;
32          }
33      }
34
    Codeium: Refactor | Explain | Generate Javadoc | ×
35      public boolean IsOperator(char c) {
36          if (c == '^' || c == '%' || c == '/' || c == '*' || c == '-' || c == '+') {
37              return true;
38          } else {
39              return false;
40          }
41      }
42
```



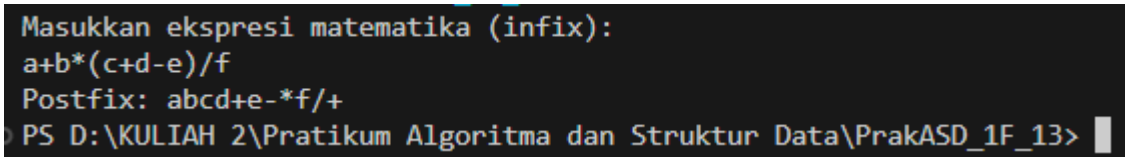
```
43      public int derajat(char c) {
44          switch (c) {
45              case '^':
46                  return 3;
47              case '%':
48                  return 2;
49              case '/':
50                  return 2;
51              case '*':
52                  return 2;
53              case '-':
54                  return 1;
55              case '+':
56                  return 1;
57              default:
58                  return 0;
59          }
60      }
61
    Codeium: Refactor | Explain | Generate Javadoc | ×
62      public String konversi(String Q) {
63          String P = "";
64          char c;
65          for (int i = 0; i < n; i++) {
66              c = Q.charAt(i);
67              if (IsOperand(c)) {
68                  P = P + c;
69              }
70              if (c == '(') {
71                  push(c);
72              }
73              if (c == ')') {
74                  while (stack[top] != '(') {
75                      P = P + pop();
76                  }
77                  pop();
78              }
79              if (IsOperator(c)) {
80                  while (derajat(stack[top]) >= derajat(c)) {
81                      P = P + pop();
82                  }
83                  push(c);
84              }
85          }
86          return P;
87      }
88  }
```

PostfixMain



```
PrakASD_1F_13 > src > P8 > PostfixMain13.java > ...
1   package P8;
2
3   import java.util.Scanner;
4
    Codeium: Refactor | Explain
5   public class PostfixMain13 {
        Run | Debug | Codeium: Refactor | Explain | Generate Javadoc | ×
6       public static void main(String[] args) {
7           Scanner sc = new Scanner(System.in);
8           String P, Q;
9           System.out.println(x:"Masukkan ekspresi matematika (infix): ");
10          Q = sc.nextLine();
11          Q = Q.trim();
12          Q = Q + ")";
13
14          int total = Q.length();
15          Postfix13 post = new Postfix13(total);
16          P = post.konversi(Q);
17          System.out.println("Postfix: " + P);
18      }
19  }
20
```

Output :



```
Masukkan ekspresi matematika (infix):
a+b*(c+d-e)/f
Postfix: abcd+e-*f/+
PS D:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

Question :

1. Pada method **derajat**, mengapa return value beberapa case bernilai sama? Apabila return value diubah dengan nilai berbeda-beda setiap case-nya, apa yang terjadi?
   Jawab : Karena untuk menyetarakan posisinya dan jika di ubah maka operator tersebut tidak akan memiliki prioritas tingkatan.

2. Jelaskan alur kerja method **konversi!**
   Jawab :
   1. String kosong diinisialisasikan lalu untuk menyimpan hasil konversi menjadi postfix
   2. Melakukan loop for melalui setiap karakter dari String Q.
   3. Jika karakter operand maka karakter ditambahkan ke String P.
   4. Jika karakter ‘(‘ maka ditambahkan ke stack.
   5. Jika karakter ‘)’ maka melakukan proses pop sampai karakter ‘(‘ ditemukan.
   6. Jika karakter operator maka melakukan pengecekan untuk menentukan urutan / prioritas operator.
   7. Lalu return P adalah profix yang berisi semua karakter dari rubahan infix ke postfix.

3. Pada method konversi, apa fungsi dari potongan kode berikut?

```
c = Q.charAt(i);
```

   Jawab : Untuk mengambil karakter pada “i” dari String Q dan disimpan di variable c.

NAMA : Gilang Purnomo
NIM : 2341720042
NO ABSEN : 13
KELAS : 1F
MATERI : STACK

2.4 Latihan Praktikum

Perhatikan dan gunakan kembali kode program pada Percobaan 1. Tambahkan dua method berikut pada class Gudang:

• Method lihatBarangTerbawah digunakan untuk mengecek barang pada tumpukan terbawah

```
91      public Barang13 lihatBarangTerbawah() {
92          if (!cekKosong()) {
93              Barang13 barangTerbawah =  tumpukan[0];
94              System.out.println("Barang terbawah: " + barangTerbawah.nama);
95              return barangTerbawah;
96          } else {
97              System.out.println(x:"Tumpukan barang kosong.");
98              return null;
99          }
100     }
101
```

Output :

```
Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Melihat barang terbawah
6. Cari barang
7. Keluar
Pilih operasi: 3
Rincian tumpukan barang di Gudang:
Kode 35: Baju (Kategori Pakaian)
Kode 24: Meja (Kategori Perabotan)
Kode 13: Raket (Kategori Alat Olahraga)

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Melihat barang terbawah
6. Cari barang
7. Keluar
Pilih operasi: 5
Barang terbawah: Raket
```

• Method cariBarang digunakan untuk mencari ada atau tidaknya barang berdasarkan kode barangnya atau nama barangnya

```
102     public Barang13 cariBarang(int kodeBarang) {
103         if (!cekKosong()) {
104             for (int i = 0; i <= top; i++) {
105                 if (tumpukan[i].kode == kodeBarang) {
106                     System.out.println("Kode Barang " + kodeBarang + " ditemukan: ");
107                     System.out.println("Nama Barang : " + tumpukan[i].nama);
108                     System.out.println("Kode Barang : " + tumpukan[i].kode);
109                     System.out.println("Kategori Barang : " + tumpukan[i].kategori);
110                     return tumpukan[i];
111                 }
112             }
113             System.out.println("Kode Barang " + kodeBarang + " tidak ditemukan.");
114             return null;
115         } else {
116             System.out.println(x:"Data pada tumpukan kosong");
117             return null;
118         }
119     }
120 }
121
```

NAMA	: Gilang Purnomo
NIM	: 2341720042
NO ABSEN	: 13
KELAS	: 1F
MATERI	: STACK

Main :

```
51          case 6:
52              System.out.print("Msukkan kode barang yang dicari: ");
53              int cariKode = input.nextInt();
54              gudang.cariBarang(cariKode);
55              break;
```

Output:

```
Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Melihat barang terbawah
6. Cari barang
7. Keluar
Pilih operasi: 6
Msukkan kode barang yang dicari: 24
Kode Barang 24 ditemukan:
Nama Barang : Meja
Kode Barang : 24
Kategori Barang : Perabotan

Menu
1. Tambah barang
2. Ambil barang
3. Tampil tumpukan barang
4. Melihat barang teratas
5. Melihat barang terbawah
6. Cari barang
7. Keluar
Pilih operasi: 7
PS D:\KULIAH 2\Pratikum Algoritma dan Struktur Data\PrakASD_1F_13>
```