Solusi UTS PILIHAN GANDA

Dasar-dasar Pemrograman 2 Kelas B-G

Semester Genap 2020/2021

Java Principles

1. Manakah pernyataan yang benar terkait Java?
    a. Semua data/nilai adalah Object.
    b. Pemeriksaan tipe dari sebuah variabel dilakukan saat proses kompilasi (sebelum program berjalan).
    c. Java memerlukan *interpreter* untuk mentranslasikan *source code* dan ke Java *byte code*.
    d. *Source code* dalam bahasa pemrograman Java tidak *portable*
2. Manakah pernyataan yang salah terkait tipe data **char**?
    a. Tipe data char mendukung skema encoding *unicode*
    b. Operator dasar aritmatika seperti + dan - dapat diterapkan pada nilai-nilai & variable bertipe char
    c. Ada beberapa karakter di tabel ASCII yang tidak dapat disimpan di sebuah variabel bertipe char.
    d. Tipe data char merupakan tipe data primitif.
3. Manakah yang merupakan *best practices* ketika memprogram dengan Java?
    a. Penamaan sebuah kelas adalah dengan mengikuti aturan *snake case*
    b. Penamaan variable harus mempunyai makna dan deskriptif.
    c. Jika sebuah variabel bertipe **int** tidak digunakan lagi, programmer perlu set isi dari variabel tersebut dengan nilai 0.
    d. Penamaan konstanta sebaiknya menggunakan *camel case*
4. Misal, ada tiga buah ekspresi yang bernilai boolean: **A**, **B**, dan **C**. Ekspresi **A** diketahui bernilai **true** dan ekspresi **B** bernilai **false**. Ekspresi **C** tidak diketahui hasil evaluasi-nya. Manakah dari pilihan berikut yang menyebabkan ekspresi **C** di-evaluasi?
    a. (A || C) && !B
    b. (A & !B) || C
    c. B && (C | A)
    d. A | (!C && B)

Basic Programming (Loops, Selections, Strings, Characters)

1. Perhatikan potongan kode berikut ini:. Berapakah nilai variabel a pada (1) dan (2)?

```
double a = 6.5;
a += a + 1;                    //(1)
a = 6;
a /= 2;                        //(2)
```

   a. (1) 14.0 dan (2) 3.0
   b. (1) 14.0 dan (2) 7.0
   c. (1) 7.5 dan (2) 3.0
   d. (1) 7.5 dan (2) 3.75

2. Perhatikan potongan kode berikut ini. Berapa kali kah string `"ddp2"` akan dicetak oleh program?

```
int var1 = 8;
int var2 = 2;
if (var1<0)
if (var2<10)
System.out.println("ddp2");
System.out.println("ddp2");
if(var1>5)
System.out.println("ddp2");
else
System.out.println("ddp2");
System.out.println("ddp2");
```

   a. 2
   b. 3
   c. 4
   d. 5

3. Manakah di antara statement berikut ini yang benar tentang keyword `break`?
   a. Harus digunakan pada `switch` statement
   b. Tidak dapat digunakan bersamaan dengan keyword `continue`
   c. Jika diletakkan di dalam loop, loop akan berhenti tanpa menjalankan statement setelahnya

d. Jika diletakkan di dalam loop, loop akan berhenti dengan menjalankan statement setelahnya

4. Perhatikan potongan kode berikut ini, apakah yang akan tercetak?

```
String a = "ohhelloeveryoneinddp2";
char[] dst = new char[10];
a.getChars(0, 5, dst, 2);
System.out.println(dst[5]);
```

a. e
b. l
c. Ohhel
d. hello

Methods & Recursion

1. Apa luaran dari potongan kode berikut?

```
public class MyClass {
    public static void main(String[] args) {
        int result = myMethod(myMethod(2, 4), myMethod(7, 7));
        System.out.println(result);
    }

    public static int myMethod(int a, int b) {
        if (a + b > 10)
            return (a + b) % 10;
        else
            return (a + b) % 7;
    }
}
```

a. 2
b. 3
c. 4
d. 5

2. Perhatikan implementasi tiga buah method berikut ini. Jika dilakukan pemanggilan method `m(3)`, berapa banyak pemanggilan yang terjadi untuk setiap method tersebut?

```
public static void m(String str) {          // Method 1
    System.out.println(str);
}

public static void m(String str, int n) {  // Method 2
    for (int i = 0; i < n; i++) {
        m(str);
    }
}

public static void m(int n) {               // Method 3
    m("Hello", n);
}
```

a. Terjadi pemanggilan Method 1 sebanyak 3 kali, Method 2 sebanyak 1 kali, Method 3 sebanyak 1 kali

b. Terjadi pemanggilan Method 1 sebanyak 1 kali, Method 2 sebanyak 3 kali, Method 3 sebanyak 1 kali

c. Terjadi pemanggilan Method 1 sebanyak 1 kali, Method 2 sebanyak 1 kali, Method 3 sebanyak 1 kali

d. Terjadi pemanggilan Method 1 sebanyak 3 kali, Method 2 sebanyak 3 kali, Method 3 sebanyak 1 kali

3. Perhatikan implementasi method berikut. Ekspresi manakah yang merepresentasikan hasil pemanggilan method `mystery(x)`?

```
public static int mystery(int n) {
    if (n == 0)
        return 1;
    else
        return 2 * mystery(n - 1);
}
```

a. Math.pow(2, x)

b. Math.pow(x, 2)

c. 2 * (x + 1)

d. x * (x + 1)

4. Pernyataan manakah yang salah mengenai method rekursif?

   a. Implementasi perulangan dalam bentuk method rekursif memerlukan relatif lebih banyak memori dibandingkan perulangan dalam bentuk iterasi biasa.

   b. Suatu method rekursif dapat mengandung satu atau lebih recursive call.

   c. Suatu method rekursif tidak selalu dapat dibuat versi iterasi non-rekursifnya.

   d. Suatu method rekursif tanpa base case beresiko menyebabkan terjadinya StackOverflowError.

Arrays

1. Manakah yang benar tentang array?
   a. Array pada Java bersifat mutable.
   b. Indeks pada array dimulai dari 1.
   c. Jumlah elemen yang dapat dimuat pada suatu array bersifat dinamis (tidak fixed).
   d. Untuk mengakses array *arr* pada indeks *i*, dapat menggunakan notasi *arr{i}*

2. Perhatikan kode berikut, apa outputnya?

```
public class Arr01{
    public static void main(String[] args){
      int[] a = {1,2,3};
      int[] b = new int[3];
      int[] c = new int[3];
      c[0] = b[0];
      c[2] = 30;
      b[1] = 20;
      b = a;
      a[0] = 10;
      System.out.print(Arrays.toString(a));
      System.out.print(Arrays.toString(b));
      System.out.print(Arrays.toString(c));
    }
}
```

a. [10, 2, 3][10, 2, 3][0, 0, 30]
b. [10, 2, 3][0, 20, 0][0, 0, 30]
c. [1, 2, 3][1, 2, 3][0, 0, 30]
d. [10, 2, 3][10, 2, 3][10, 2, 3]

3. Manakah inisialisasi array berikut yang salah?

a. Benar semua
b. Object[] a = new Object[]{1,"2",'3'};
c. int[][] b = new int[2][3];
d. char[][] c = new char[][]{{'a',"2".charAt(0)}};

4. Method berikut dimaksudkan untuk mencari rata-rata elemen pada array 2D. Jika terdapat bug, manakah baris kode yang mengandung bug?

```
1  public static double f(double[][] arrOfArrs) {
2        int ttl = 0;
3        int cnt = -5;
4        for(double[] arr:arrOfArrs) {
5              for(int i=1; i<arr.length; i++) {
6                    ttl += arr[i];
7                    cnt += (int) 1.5;
8              }
9        }
10       return (double) ttl/(cnt+5);
11 }
```

a. Baris 5
b. Tidak terdapat bug.
c. Baris 3
d. Baris 7

OOP

Perhatikan potongan kode berikut ini untuk soal OOP (1) - (4)

```
public class MainClass {

    // Program untuk menyapa sahabat
    Public static void main(String[] args){
       // (2) mencetak intro
       // (3) membuat object baru bernama myObject
       // (4) memperbaharui sapaan
       // (5) mencetak sapaan
    }
}
class MyClass{

    private int numOfItems;
    private String reportTitle;

    // (1) constructor dari MyClass

    public static void intro(){
       // mencetak intro
    }

    public void update(int num, String title){
       // memperbaharui sapaan
    }

    public void print(){
       // mencetak sapaan
    }
}
```

1. Kamu akan mendefinisikan constructor dari `MyClass` (label: `//(1)`). Mana di antara pernyataan berikut yang tepat?
    a. `public MyClass`
    b. `public void MyClass`
    c. `public MyClass()`
    d. `public void MyClass()`

2. Kamu akan memanggil method yang mencetak intro (label: `//(2)`). Mana di antara pernyataan berikut yang tepat?
    a. `MainClass.intro();`
    b. `myObject.intro();`
    c. `MyClass.intro();`
    d. `intro();`
3. Kamu akan membuat object baru dari `MyClass` (label: `//(3)`). Mana di antara pernyataan berikut yang tepat?
    a. `MyClass myObject;`
    b. `MyClass myObject = MyClass();`
    c. `MyClass myObject = new MyClass();`
    d. `MyClass myObject = new MyClass(MyClass);`
4. Kamu akan memanggil method update (label: `//(4)`). Mana di antara pernyataan berikut yang tepat?
    a. `update(myObject(3, "Apa Kabar Kawan?"));`
    b. `update(3, "Apa Kabar Kawan?");`
    c. `myObject.update(3, "Apa Kabar Kawan?");`
    d. `MyClass.update(3, "Apa Kabar Kawan?");`

Solusi UTS ESAI

Dasar-dasar Pemrograman 2 Kelas B-G

Semester Genap 2020/2021

Methods with Array

1. Perhatikan implementasi method berikut.

```
// precondition: x != 0
public static void doSomething(int[] arr, int x) {
    int i = 0;
    int j = 0;
    while (i < arr.length) {            // Loop 1
        if (arr[i] == x) {
            i++;
        } else {
            arr[j++] = arr[i++];
        }
    }
    while (j < arr.length) {            // Loop 2
        arr[j++] = 0;
    }
}
```

   a. [2 poin] Jika diketahui sebuah array `int[] myArr = {3, 1, 2, 2, 1}`, bagaimana isi dari `myArr` setelah dilakukan pemanggilan `doSomething(myArr, 1)`?

      Jawab: {3, 2, 2, 0, 0}

   b. [2 poin] Jika diketahui sebuah array `int[] myArr = {1, 1, 1}`, bagaimana isi dari `myArr` setelah dilakukan pemanggilan `doSomething(myArr, 1)`?

      Jawab: {0, 0, 0}

   c. [4 poin] Secara umum, apa yang dilakukan oleh method `doSomething`?

      Jawab: Method `doSomething` menghapus elemen yang bernilai `x` dari array `arr`, menggeser elemen lainnya (yg bukan `x`) ke indeks yang lebih kecil, dan mengisi slot yang kosong dengan 0.

d. [4 poin] Kondisi seperti apa yang menyebabkan bodi pada Loop 2 tidak pernah dieksekusi?

Jawab: Bodi Loop 2 tidak dieksekusi saat tidak ditemukan elemen `x` di dalam array `arr`.

Methods with 2D Array

Kak Burhan membuat sebuah program untuk mengecek jadwal mahasiswa. Jadwal ini disimpan dalam bentuk matriks dimana setiap barisnya merepresentasikan slot kuliah dan setiap kolomnya merepresentasikan hari.

```
String[][] jadwalMhs1 = {{"DDP2","DDAK", "DDP2", "DDAK", "PPSI"},
                         {"DDP2", "DDAK", "DDP2", "DDAK", "PPSI"},
                         {null, null, null, null, null},
                         {"MPKT", "DDP2", null, null, null},
                         {"MPKT", "DDP2", null, null, null}};

String[][] jadwalMhs2 = {{"DDP2","DDAK", "DDP2", "DDAK", "MPKT"},
                         {"DDP2", "DDAK", "DDP2", "DDAK", "MPKT"},
                         {null, null, null, null, null},
                         {"PPSI", "DDP2", null, null, null},
                         {"PPSI", "DDP2", null, null, null}};
```

Salah satu method yang dibuat oleh kak Burhan menerima input jadwal dari dua mahasiswa dan mengembalikan daftar mata kuliah yang sama.

```
public static String[] cekKesamaan(String[][] a, String[][] b){

   String[] matkulSama = {};

   for(int i = 0; i < a.length; i++){
     for(int j = 0; j < a[i].length; j++){
       if(a[i][j].equals(b[i][j])){
         matkulSama = Arrays.copyOf(matkulSama, matkulSama.length+1);
         matkulSama[matkulSama.length-1] = a[i][j];
       }
     }
```

```
    }
    return matkulSama;
  }
```

Jika `cekKesamaan(a,b)` menerima argumen `jadwalMhs1` dan `jadwalMhs2`, ekspektasi outputnya adalah `["DDP2","DDAK", "PPSI", "MPKT"]`.

Ternyata method `cekKesamaan(a,b)` masih memiliki banyak bugs.

[4 poin] Sebutkan dua bugs yang mungkin dimiliki oleh method `cekKesamaan(a,b)`!

[8 poin] Perbaiki dua bugs pada method `cekKesamaan(a,b)` yang sudah kamu identifikasi dari pertanyaan sebelumnya !

Jawab:

1. Null Pointer Exception pada blok IF.
2. `matkulSama` berisi daftar mata kuliah yang berulang.
3. PPSI dan MPKT tidak terdeteksi sebagai mata kuliah yang sama.

```
public static String[] cekKesamaan(String[][] a, String[][] b){

   String[] matkulSama = {};

   //iterasi seluruh elemen array pertama
   for(int i = 0; i < a.length; i++){
     for(int j = 0; j < a[i].length; j++){
       if(a[i][j] != null){ //null handler
         //iterasi seluruh elemen array kedua
         for(int k = 0; k < b.length; k++){
           for(int l = 0; l < b[k].length; l++){
             if(b[k][l] != null && b[k][l].equals(a[i][j])){
              //pengecekan duplikasi elemen array matkulSama
               if(!Arrays.toString(matkulSama).contains(b[k][l])){
                 matkulSama = Arrays.copyOf(matkulSama,
                  matkulSama.length+1);
                 matkulSama[matkulSama.length-1] = b[k][l];
               }
             }
           }
```

```
            }
        }
    }
}
return matkulSama;
}
```

Classes & Objects (1)

Saat ini kita sudah memasuki bulan Ramadhan! Kak Burhan ingin mengadakan bakti sosial melalui distribusi sembako ke orang-orang yang membutuhkan. Untuk itu, Kak Burhan membuat class Kardus dan class Sembako yang menggambarkan kardus sumbangan yang dapat diisi dengan sembako.

Untuk mencegah terbentuknya banyak sampah, kardus yang digunakan adalah berbagai kardus bekas yang dapat menampung berat maksimum yang berbeda-beda. Dengan begitu, kombinasi sembako dalam setiap kardus dapat berbeda-beda tergantung berat maksimum kardus, dan juga ketersediaan sembako di warung Pacil Oke.

Saat ini, Kak Burhan sudah menyiapkan 2 buah kardus yang siap didistribusikan, myKardus1 dan myKardus2.

Perhatikan class DistribusiSembako, Kardus, dan Sembako berikut ini.

```
import java.util.Scanner;

public class DistribusiSembako{
  public static void main(String[] args){
      //kardus 1
      Kardus myKardus1 = new Kardus(6);
      myKardus1.addSembako(new Sembako("gula", 0.5, 9000));
      myKardus1.addSembako(new Sembako("minyak", 1, 18000));
      myKardus1.addSembako(new Sembako("susu", 1, 32000));
      myKardus1.addSembako(new Sembako("tepung", 1, 21000));
      myKardus1.addSembako(new Sembako("garam", 0.5, 7000));

      //kardus 2
```

```
      Kardus myKardus2 = new Kardus(8);
      myKardus2.addSembako(new Sembako("gula", 1, 16000));
      myKardus2.addSembako(new Sembako("susu", 1.5, 22000));
      myKardus2.addSembako(new Sembako("beras", 10, 110000));
      myKardus2.addSembako(new Sembako("gula", 2, 40000));
      myKardus2.addSembako(new Sembako("minyak", 1.5, 25000));


       //cetak kardus
      System.out.println(myKardus1);
      System.out.println(myKardus2);

      //check kardus
      if(myKardus1.equals(myKardus2))
        System.out.println("Kedua kardus beratnya sama!");
      else
        System.out.println("Kedua kardus beratnya beda!");
  }
}
```

```
class Kardus{
  private double kapasitasKardus;
  private double beratKardus;
  private double nilaiKardus;
  private Sembako[] listSembako;
  private int jumlahSembako;

  Kardus(double kapasitasKardus){
      this.setKapasitas(kapasitasKardus);
      this.listSembako = new Sembako[5];
      this.beratKardus = 0;
      this.jumlahSembako = 0;
      this.nilaiKardus = 0;
  }

  public void setKapasitas(double kapasitasKardus){
      this.kapasitasKardus = kapasitasKardus;
  }

  public void addSembako(Sembako newSembako){
      if(masihMuat(newSembako.getBerat()) && !sudahAda(newSembako)){
        listSembako[jumlahSembako++] = newSembako;
        beratKardus += newSembako.getBerat();
        nilaiKardus += newSembako.getHarga();
      }
  }

  public boolean masihMuat(double beratTambahan){
      return this.beratKardus + beratTambahan <
this.kapasitasKardus;
  }

  public boolean sudahAda(Sembako newSembako){
      boolean ada = false;
      for(int i=0;i<this.jumlahSembako;i++){
        if(listSembako[i].equals(newSembako))
            ada = true;
      }
      return ada;
  }

  public double getBerat(){
      return this.beratKardus;
  }
```

```
  public boolean equals(Kardus otherKardus){
      return this.beratKardus == otherKardus.getBerat();
  }
//(1)



}
```

```
class Sembako{
  private String nama;
  private double berat;
  private double harga;

  Sembako(String nama, double berat, double harga){
      this.nama = nama;
      this.berat = berat;
      this.harga = harga;
  }

  public String getNama(){
      return this.nama;
  }

  public double getBerat(){
      return this.berat;
  }

  public double getHarga(){
      return this.harga;
  }

  public boolean equals(Sembako otherSembako){
      return otherSembako.getNama().equals(this.nama);
  }

  public String toString(){
      return String.format("%s (%.2f kg): Rp.%.2f", this.nama,
this.berat, this.harga);
  }
}
```

```
//(2) Output yang diharapkan:
Kardus (5 barang) seberat 4.00 kg senilai Rp.87000.00
Kardus (_____ barang) seberat _____ kg senilai Rp._______
//(a,b,c)
Kedua kardus beratnya _______!                                  //(d)
```

1. [4 poin] Lengkapilah class Kardus dengan method yang dibutuhkan untuk menjalankan program ini! (Hint: perhatikan method main dan luaran yang diharapkan)

```
  public String toString(){
      return String.format("Kardus (%d barang) seberat %.2f kg
senilai Rp.%.2f", this.jumlahSembako, this.beratKardus,
this.nilaiKardus);
   }  //atau variasi sejenis
```

2. [2 poin] Lengkapilah output yang diharapkan pada poin (a), (b), (c) dan (d)! **Pisahkan jawaban anda dengan koma!**

   3,4.00,63000.00,sama

```
Kardus (5 barang) seberat 4.00 kg senilai Rp.87000.00
Kardus (3 barang) seberat 4.00 kg senilai Rp.63000.00     //(a,b,c)
Kedua kardus beratnya sama!                                //(d)
```

3. [6 poin] Jelaskan proses anda mendapatkan jawaban (2).

   Dalam mengisi `myKardus1`, terdapat 5 obyek `Sembako` yang ditambahkan melalui `addSembako()`, dan semuanya dapat ditambahkan karena semuanya sesuai kapasitas `myKardus1` (melalui pengecekan `masihMuat()`), dan tidak ada pengulangan (melalui pengecekan `sudahAda()`), sehingga semuanya ditambahkan. Hasil akhir `myKardus1` berisi `5` barang sebesar `4.00` kg total senilai `Rp.87000.00`.

   Dalam mengisi `myKardus2`, terdapat 5 obyek `Sembako` yang ditambahkan `addSembako()`, tetapi tidak semua berhasil ditambahkan. Beras tidak ditambahkan karena melebihi kapasitas `myKardus2` (`masihMuat()` bernilai `false`), dan gula yang kedua tidak ditambahkan karena sudah ada gula di dalam `myKardus2` (`sudahAda()` bernilai `false`). Hasil akhir `myKardus2` berisi `3` barang sebesar `4.00` kg total senilai Rp.`63000.00`. **(a)-(c)**.

Pada akhirnya, dilakukan pengecekan *conditional* `myKardus1.equals(myKardus2)`. Method `equals()` pada kelas `Kardus` hanya memeriksa apakah `beratKardus` kedua kardus sama. Karena kedua kardus beratnya 4.00 kg, maka method `equals()` bernilai `true`, dan tercetak "`Kedua kardus beratnya sama`!" **(d)**

Classes & Objects (2)

Kak Burhan ingin mengembangkan program untuk pengelolaan soal-soal kuis. Dua buah kelas yang dibutuhkan untuk permasalahan tersebut adalah kelas **Question** dan kelas **Quiz**.

Kelas **Question** merepresentasikan sebuah pertanyaan dan mempunyai dua buah informasi penting, yaitu **description** (deskripsi dari persoalan tersebut) dan **score** (nilai maksimum yang diberikan jika peserta menjawab pertanyaan tersebut dengan benar).

Kelas **Quiz** berisi **"array of Questions"**, dan mempunyai beberapa method untuk pengelolaan soal, seperti **addQuestion(Question q)** untuk menambahkan sebuah soal kedalam sebuah object **Quiz**.
Silakan Anda perhatikan kode sumber berikut:

```
class Question {
  private String desc;
  private int score;

  public Question(String desc, int score) {
    this.desc = desc;
    this.score = score;
  }

  public String getDesc() {
    return this.desc;
  }

  public int getScore() {
    return this.score;
  }

  public String cetak() {
```

```
    return "Question = " + this.desc + "\t Score = " + this.score;
  }
}

class Quiz {
  private Question[] questions;
  private int numQuestions;

  private static final int DEFAULT_NUM_QUESTIONS = 10;

  public Quiz() {
    this.questions = new Question[DEFAULT_NUM_QUESTIONS];
    this.numQuestions = 0;
  }

  public void addQuestion(Question q) {
    this.questions[this.numQuestions++] = q;
  }

  //Menghapus question pertama yang ada di array of Questions
  public void removeFirstQuestion() {

    // Implementasikan method ini!

  }

  public String cetak() {
    StringBuffer outStr = new StringBuffer();
    for (int i = 0; i < this.numQuestions; i++) {
      outStr.append(this.questions[i].cetak() + "\n");
    }
    return outStr.toString();
  }
}

public class Test {
  public static void main(String[] args) {
    Quiz quiz = new Quiz();

    quiz.addQuestion(new Question("Apa itu Variable?", 10));
    quiz.addQuestion(new Question("Apa itu Unboxing?", 15));
    quiz.addQuestion(new Question("Apa itu Garbage Collector?", 15));
```

```
    quiz.removeFirstQuestion(); // hapus question pertama

    System.out.println(quiz.cetak());
    // Mencetak:
    // Question = Apa itu Unboxing  Score = 10
    // Question = Apa itu Garbage Collector  Score = 10
  }
}
```

[6 poin] Apakah kelas Quiz dan kelas Question bersifat Immutable? Mengapa?

Jawab:

Kelas Quiz tidak immutable karena state dari sebuah object Quiz dapat berubah-ubah. Sebagai contoh, banyaknya pertanyaan yang ada di dalam sebuah Quiz dapat bertambah dan berkurang.

Kelas Question bersifat immutable karena semua instance variables bersifat private, dan tidak ada method yang dapat digunakan untuk mengubah state/nilai dari salah satu instance variable tersebut. Yang paling penting, tidak ada method yang mengembalikan alamat ke object yang mutable.

Seandainya ada yang menjawab dengan cara di bawah, juga akan dibenarkan:
Keduanya bersifat mutable KARENA kedua kelas tersebut tidak bersifat FINAL.
Namun, nilai akan -1 jika menambahkan statement bahwa instance variable-nya harus FINAL juga. Untuk menjadi immutable, instance variable tidak harus bersifat FINAL.

[6 poin] Perhatikan kembali kelas Quiz. Method **removeFirstQuestion()** digunakan untuk menghapus pertanyaan pertama pada **array of Questions** yang ada di object Quiz. Anda juga dapat melihat contoh penggunaan method ini di program utama (kelas **Test**). Bagaimana implementasi dari method ini?

Cukup tuliskan *method header* dan *method body*-nya saja, tanpa perlu tulis ulang semua isi yang ada di kelas Quiz.

Jawab:

```
public void removeFirstQuestion() {
```

```
    for (int i = 0; i < this.numQuestions - 1; i++) {
      this.questions[i] = this.questions[i+1];
    }
    this.questions[--this.numQuestions] = null;
}
```

Methods & Recursion

Perhatikan dua kode sumber berikut.

```
// Person.java
public class Person {
  String name;
  private Person child;
  Person(String name) { this.name = name; }
  Person getChild() { return this.child; }
  void setChild(Person child) { this.child = child; }
  public String toString() { return this.name; }
}

// PersonTest.java
public class PersonTest {
  public static void main(String[] args) {
    Person b = new Person("Bob");
    Person c = new Person("Cindy");
    Person d = new Person("Donna");
    Person e = new Person("Emma");
    Person f = new Person("Finna");
    b.setChild(c);
    c.setChild(d);
    e.setChild(f);
    System.out.println(b.getYoungestInFamily()); // Donna
    System.out.println(e.getYoungestInFamily()); // Finna
    System.out.println(f.getYoungestInFamily()); // Finna
  }
}
```

Jawablah pertanyaan-pertanyaan berikut.

- [2 poin] Sebutkan dan jelaskan perbedaan antara access modifier untuk field name dan child pada Person.java.

Jawab:

Field name memiliki access modifier default: visible only within its own package.

Field child memiliki access modifier private: can only be accessed in its own class.

- [5 poin] Tambahkan method getYoungestInFamily() pada Person.java dengan kriteria sbb:
    - Method tsb dimaksudkan untuk mendapatkan person termuda dari suatu "family" yang terbentuk via relasi child (dan child of child, serta child of child of child, dan seterusnya). Jika ternyata person tersebut tidak memiliki child, person termudanya dalam family adalah dia sendiri. Untuk lebih jelasnya, perhatikan contoh pada PersonTest.java.
    - Method tsb mengembalikan suatu object bertipe Person.
    - Pastikan method tersebut dapat dipanggil setidaknya dari class lain di package yang sama.
    - Method tersebut wajib menggunakan rekursi. Pengerjaan dengan iterasi akan memperoleh 0 poin.

    Jawab:

```
Person getYoungestInFamily() {

    // base case
    if (this.child == null)
        return this;
    // recursive case
    else
        return this.child.getYoungestInFamily();

}
```

[5 poin] Beri penjelasan secara rinci base case, recursive case, serta cara kerja kode yang Anda tambahkan untuk method getYoungestInFamily().

Jawab:

Base case: Base case merupakan kasus dasar untuk dapat keluar dari proses rekursi. Biasa disebut juga dengan terminating condition. Dalam method tersebut, base case-nya adalah ketika person tsb tidak memiliki child (**this.child == null**), yang berarti dia sendiri adalah person termuda, dan method akan mengembalikan person itu sendiri.

Recursive case: Recursive case merupakan kasus yang mengandung pemanggilan method itu sendiri (rekursi), dan tentunya harus dipastikan nilai argumennya semakin mendekati ke base case. Dalam method tsb, recursive case-nya adalah ketika person tsb memiliki child (bagian **else**), sehingga akan dipanggil lagi method tsb tapi kali ini untuk child dari person tsb.

Cara kerja method tsb adalah, jika person tsb tidak memiliki anak (= yang berarti dia termuda), maka akan masuk ke base case dan mengembalikan person itu sendiri. Akan tetapi, jika person tsb memiliki anak (= yang berarti dia bukan termuda), maka akan masuk ke recursive case dan mengembalikan anggota keluarga termuda (via getYoungestInFamily()) dari anak si person tsb.