Modul 6
Perulangan Bagian Pertama

Petunjuk

1. Perintah untuk mengumpulkan berkas kode program dan jawaban akan dituliskan dalam warna merah.
2. Pertanyaan yang harus Anda jawab akan dituliskan dalam warna biru muda dan memiliki nomor. Tulis jawabannya pada sebuah berkas teks dengan nama M0601xxyyy.txt dengan xx adalah tahun angkatan dan yyy adalah nomor urut mahasiswa.
3. Anda diperkenankan membuka materi, kecuali pada saat tes awal dan kuis.

Pada minggu lalu, Anda sudah mempelajari dan mengimplementasikan struktur kendali perkondisian. Dalam kegiatan praktikum minggu ini, Anda akan mempelajari dan mengimplementasikan struktur kendali perulangan dengan perintah `while`, `do while`, dan `for`. Sebelum itu, Anda akan mempelajari tipe data *String* dan *class* Random.

Seluruh *method* yang diminta sebagai bagian dari modul ini akan dibuat pada *class* dengan nama DemoPerulangan. *Class* ini akan kita gunakan untuk membantu Anda berlatih menggunakan perintah *while*, oleh karena itu, Anda tidak perlu memperhatikan konstruksi dan perancangan *class*-nya.

A. *String*

String merupakan salah satu tipe data *object reference* yang disediakan oleh Java. *String* berbentuk untaian karakter. Berikut ini adalah beberapa contoh deklarasi *string*.

```
String contohString1 = new String();
String contohString2 = "Saya suka Java";
```

Perintah *new String*() merupakan sebuah constructor *string* yang mengisinya dengan *string* kosong (""). Pada deklarasi pertama, kita membuat obyek *string* baru dan mengisinya ke variabel contohString1. Pada deklarasi kedua, kita mempunyai literal *string* dan mengisinya ke variabel contohString2.

Java menyediakan beberapa *method* untuk tipe data ini. Berikut ini disajikan beberapa *method* disertai penjelasannya.

1. `public int length()`
   Method ini mengembalikan panjang dari *string*.
2. `public boolean isEmpty()`
   Method ini mengembalikan nilai *true*, jika panjang *string* adalah 0.
3. `public char charAt(int index)`
   Method ini mengembalikan sebuah karakter. Karakter yang dikembalikan terletak pada posisi ke-index. Catatan: posisi dihitung mulai dari 0.
4. `public boolean equals(Object anObject)`
   Method ini membandingkan *string* dengan sebuah obyek. *Method* ini mengembalikan nilai *true* jika dan hanya jika keduanya memiliki untaian karakter yang sama.
5. `public boolean equalsIgnoreCase(String anotherString)`
   Method ini membandingkan *string* sekarang dengan *string* lain dengan mengabaikan perbedaan antara karakter huruf besar dan huruf kecil. *Method* ini mengembalikan nilai *true* jika dan hanya jika keduanya memiliki untaian karakter yang sama dengan mengabaikan perbedaan huruf besar dan kecil.
6. `public int compareTo(String anotherString)`
   Method ini membandingkan *string* sekarang dengan *string* lain secara leksikografis. *Method* ini mengembalikan nilai selisih antara *string* sekarang dan *string* lain. Selisih dihitung berdasarkan karakter-karakter pada posisi yang sama.

Untuk lebih memahami String, buatlah sebuah *class* bernama LatihanString dan tuliskan kode berikut. Kemudian jawablah pertanyaan di bawahnya.

```
public int hitungPanjangString(String aString){
    return aString.length();
}

public boolean periksaStringKosong(String aString){
    return aString.isEmpty();
}

public char cariKarakter(String aString, int index){
    return aString.charAt(index);
}

public boolean cekStringSama(String aString, String bString){
    return aString.equals(bString);
}

public boolean cekStringSama2(String aString, String bString){
    return aString.equalsIgnoreCase(bString);
}

public int perbandinganString(String aString, String bString){
    return aString.compareTo(bString);
}

public String konkatenasiString(String aString, String bString){
    return aString+bString;
}
```

Gambar 1. Kode Program untuk Latihan *String*.

1. Berapakah nilai kembalian dari *method* hitungPanjangString jika diberi parameter “Wombat”?
2. Apakah nilai kembalian dari *method* periksaStringKosong jika diberi parameter “”?
3. Apakah nilai kembalian dari *method* cariKarakter jika diberi parameter “Wombat” dan 5?
4. Apakah nilai kembalian dari *method* cekStringSama jika diberi parameter “Wombat1” dan “Wombat2”?
5. Apakah nilai kembalian dari *method* cekStringSama2 jika diberi parameter “Wombat1” dan “WOMBAT1”?
6. Berapakah nilai kembalian dari *method* perbandinganString jika diberi parameter “a” dan “”?
7. Berapakah nilai kembalian dari *method* perbandinganString jika diberi parameter “” dan “a”?
8. Berapakah nilai kembalian dari *method* perbandinganString jika diberi parameter “a” dan “b”?
9. Berapakah nilai kembalian dari *method* perbandinganString jika diberi parameter “b” dan “a”?
10. Berapakah nilai kembalian dari *method* perbandinganString jika diberi parameter “a” dan “ab”?
11. Berapakah nilai kembalian dari *method* perbandinganString jika diberi parameter “ab” dan “a”?
12. Apakah nilai kembalian dari *method* konkatenasiString jika diberi parameter “Praktikum” dan “PBO”?

Untuk diperhatikan bahwa posisi karaker dalam *string* dimulai pada angka 0. Dengan demikian, karakter terakhir dari *string* terletak pada posisi (panjangkarakter-1). Sementara itu, operator “+” terhadap *string* menunjukkan operasi konkatenasi *string*, yaitu menempelkan sebuah *string* ke *string* lain. Kumpulkan berkas *project* ini dengan nama M0601xxyyy.jar.

B. *Random Number*

Kita akan mempelajari *random number* (bilangan acak). Java menyediakan sebuah *class* bernama Random yang digunakan untuk membangkitkan deretan nilai acak. Nilai acak yang dibuat bukanlah benar-benar acak, melainkan dibangkitkan dengan rumus sehingga seolah-olah seperti acak. *Class* random tidak hanya mampu membuat nilai acak untuk *integer*, tetapi juga meliputi *boolean*, *long*, *float*, dan *double*. Java menggunakan sebuah "benih" ketika membangkitkan nilai acak. Berikut ini adalah deklarasi untuk nilai acak.

```
Random rand = new Random();
```

Perintah *new Random*() akan membuat sebuah pembangkit nilai acak. Kemudian hasilnya akan disimpan ke variabel rand. Dari rand, kita dapat menciptakan bilangan-bilangan acak yang kita inginkan.

Class Random menyediakan beberapa *method* yang dapat mempermudah kita dalam membuat bilangan acak. Berikut ini disajikan beberapa *method* tersebut disertai dengan penjelasannya.

1. `public boolean nextBoolean()`
   Method ini mengembalikan sebuah *boolean* acak yang distribusinya *uniform*.
2. `public double nextDouble()`
   Method ini mengembalikan sebuah *double* acak yang distribusinya *uniform* dengan rentang nilai 0,0 sampai 1,0.
3. `public float nextFloat()`
   Method ini mengembalikan sebuah *float* acak yang distribusinya *uniform* dengan rentang nilai 0,0 sampai 1,0.
4. `public int nextInt()`
   Method ini mengembalikan sebuah *integer* acak yang distribusinya *uniform* berdasarkan deret pembangkit bilangan acak.
5. `public int nextInt(int n)`
   Method ini mengembalikan sebuah *integer* acak yang distribusinya *uniform* dengan rentang nilai $[0, n)$. Dengan kata lain, rentangnya adalah 0 sampai $n - 1$.
6. `public long nextLong()`
   Method ini mengembalikan sebuah *long* acak yang distribusinya *uniform* berdasarkan deret pembangkit bilangan acak.

Untuk lebih memahami *class* Random, buatlah sebuah *class* bernama LatihanRandom dan tuliskan kode berikut. Kemudian jawablah pertanyaan di bawahnya.

```
import java.util.Random;

public class LatihanRandom{
    private Random rand;

    public LatihanRandom(){
        this.rand = new Random();
    }

    public int getRandom(){
        return this.rand.nextInt(5)+1;
    }

    public double getRandom2(){
        return 100.0+(this.rand.nextDouble()*(1000.0-100.0));
    }
}
```

Gambar 2. Kode Program untuk Latihan Random.

13. Berapakah rentang nilai kembalian untuk *method* getRandom?
14. Berapakah rentang nilai kembalian untuk *method* getRandom2?

Untuk menggunakan *class* Random, jangan lupa tambahkan perintah `import java.util.Random` di *header* program Anda. Kode tersebut akan memanggil *class* Random yang dibutuhkan oleh program Anda. Penjelasan tentang *import* akan dibahas lebih mendalam dalam materi *package* yang akan diajarkan setelah ujian tengah semester. Kumpulkan berkas *project* ini dengan nama M0602xxyyy.jar.

C. *While* Bagian Pertama

Skema perulangan *while* biasanya digunakan untuk **melakukan perulangan yang kita belum ketahui dengan pasti berapa kali perulangan yang akan dilakukan**, misalnya:

1. Memeriksa semua bilangan positif untuk mencari angka paling kecil yang merupakan kelipatan 4 dan 10.
2. Mencari nilai n yang paling kecil dan memenuhi syarat apabila kita menjumlahkan semua *integer* positif dari 1 sampai dengan n $(1 + 2 + 3 + \ldots + n)$, maka nilai totalnya akan lebih besar dari 40.

Skema perulangan *while* adalah skema yang paling dasar dan sebenarnya semua skema perulangan yang lain dapat dibuat dengan menggunakan skema ini. Perhatikan sintaks *while* di bawah ini.

```
while (kondisi)
     statement(s)
```

Keterangan:

- **kondisi**: ekspresi *boolean* yang nilainya akan menentukan apakah bagian ***statement*(*s*)** akan dikerjakan atau tidak.
- ***statement*(*s*)**: jika **kondisi** bernilai `true`, maka ***statement*(*s*)** akan dieksekusi . Dengan kata lain, program akan masuk ke bagian perulangan dan mengeksekusi semua *statement* yang ada di dalamnya. Jika kondisi bernilai `false`, maka ***statement*(*s*)** tidak akan dieksekuisi. Dengan kata lain, *method* akan langsung melanjutkan ke *statement* berikutnya setelah skema *while*.
- Setiap kali program telah menyelesaikan bagian ***statement*(*s*)**, program akan memeriksa lagi nilai **kondisi**. Jika **kondisi** masih bernilai `true`, program akan mengeksekusi kembali ***statement*(*s*)**. Hal ini dilakukan terus sampai suatu saat ketika nilai **kondisi** adalah `false`.
- Perhatikan bahwa pada saat awal skema ini akan langsung memeriksa nilai **kondisi** sebelum memasuki ***statement*(*s*)**.

15. Berapa kalikah jumlah eksekusi minimal ***statement*(*s*)** dari skema *while*? Petunjuk: jumlah eksekusi minimal akan terjadi apabila **kondisi** bernilai `false` pada saat *while* mulai dijalankan.

Sebagai contoh awal, Anda akan menggunakan skema *while* untuk menyelesaikan masalah berikut ini. Carilah kelipatan paling kecil dari bilangan bulat x yang lebih besar dari bilangan bulat y. Asumsikan bahwa x dan y adalah bilangan bulat yang lebih besar atau sama dengan 0. Sebagai contoh, jika x adalah 3 dan y adalah 16, maka jawabannya adalah 18.

Ide dasarnya adalah kita akan memeriksa kelipatan x dimulai dari x. Kita akan melakukan hal ini sampai kelipatan tersebut lebih dari y. Jika kelipatan tersebut tidak lebih besar dari y, maka tambahkan x ke kelipatan tersebut sehingga didapatkan kelipatan x berikutnya. Berikut ini adalah potongan kode program untuk menyelesaikan masalah di atas.

```
public int demoKelipatan(int kelipatan, int batas){
    int hasil=kelipatan;
    while(hasil<=batas){
        hasil=hasil+kelipatan;
    }
    return hasil;
}
```

Gambar 3. Potongan Kode Program demoKelipatan.

Biasakanlah untuk selalu menggunakan blok *statement* pada saat kita membuat perulangan walaupun kita hanya ingin menjalankan satu buah ekspresi. Alasannya sama dengan alasan mengapa kita sebaiknya selalu menggunakan blok *statement* pada saat kita menggunakan perintah *if*.

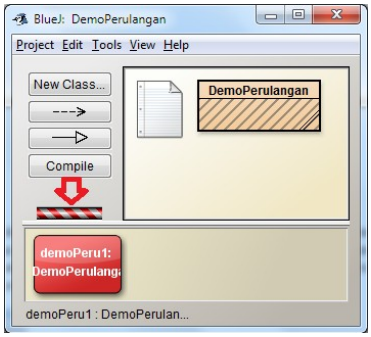
16. Bila kelipatan= 7 dan batas=5 maka bagian *statement* dari skema *while* ini akan dieksekusi sebanyak berapa kali?

17. Bila kelipatan= 0 dan batas=2 maka bagian statement dari skema *while* ini akan dieksekusi sebanyak berapa kali?

D. *Infinite Loop* dan BlueJ

Dalam proses belajar perulangan, suatu hari Anda hampir pasti akan membuat kesalahan yang menyebabkan program Anda melakukan perulangan tanpa henti (*infinite loop*). Sebagai contoh, program di atas mengandung kesalahan dimana dia akan melakukan perulangan tanpa henti bila kita memasukkan kelipatan dengan nilai 0. Karena itu, kita perlu mengetahui caranya me-*reset Java Virtual Machine* yang digunakan BlueJ agar kita dapat menghentikan *infinite loop* tersebut.

Panggillah *method* demoKelipatan tersebut dengan parameter kelipatan bernilai 0 dan batas bernilai 5. Tidak seperti biasanya, BlueJ tidak akan langsung memberikan jawabannya. Perhatikan juga bahwa di atas daftar objek-objek akan ada garis-garis merah yang bergerak horizontal. Garis-garis tersebut menandakan bahwa program Anda masih sedang berjalan pada saat ini.



Gambar 4. *Infinite Loop* pada BlueJ.

Untuk me-*reset* JVM yang digunakan BlueJ, klik kanan pada garis merah tersebut, dan pilih *Reset Java Virtual Machine*. Perbaikilah *method* demoKelipatan agar dia mengembalikan nilai -1 bila nilai parameter kelipatan adalah 0.

E. *While* Bagian Kedua

Sekarang, kita akan menggunakan perulangan untuk masalah yang sedikit lebih kompleks. Sebagai contoh kasus, kita akan menggunakan masalah di bawah ini :

Buatlah sebuah *method* di *class* DemoPerulangan dengan nama kelipatan3Menaik. *Method* ini akan menerima dua buah *integer*, yaitu batasBawah dan batasAtas. *Method* ini akan mengembalikan sebuah objek *string* yang berisi kelipatan-kelipatan 3 mulai dari batasBawah sampai dengan batasAtas. Tiap kelipatan 3 ini akan dipisahkan oleh sebuah spasi, dan dituliskan secara terurut menaik. Sebagai contoh, jika batasAtas adalah 3 dan batasBawah adalah 15, maka *method* ini akan mengembalikan objek *string* bernilai "3 6 9 12 15". Kode dari *method* tersebut dapat dilihat di bawah ini :

```
public String kelipatan3Menaik(int batasBawah, int batasAtas){
    String result="";
    int currentNumber = batasBawah;
    while(currentNumber<=batasAtas){
        if(currentNumber%3==0){
            result=result+currentNumber+" ";
        }
        currentNumber++;
    }
    return result.trim();
}
```

Gambar 5. Potongan Kode Program Kelipatan3Menaik.

18. Apa fungsi dari *method* `trim` pada kode di atas ?
19. Apa isi objek *String* yang dikembalikan bila batasBawah>batasAtas? Apakah hasil ini sesuai dengan spesifikasi yang diminta? Jelaskan.

Buatlah sebuah *method* dengan nama kelipatan3Menurun yang juga akan mengembalikan *string* berisi sejumlah bilangan kelipatan 3 dari batas atas sampai dengan batas bawah namun secara terurut menurun. Kumpulkan berkas *project* ini dengan nama M0603xxyyy.jar.

F. *Do While*

Skema *do while* adalah skema perulangan yang sangat mirip dengan skema *while*. Perbedaannya adalah bahwa kondisi pengulangan pada skema *do while* baru diperiksa **setelah** *statement*/blok *statement* miliknya selesai dieksekusi.

```
do
   statement(s)
while (kondisi)
```

20. Berapa kalikah jumlah eksekusi minimal *statement*(*s*) dari skema *do while*? Petunjuk: jumlah eksekusi minimal akan terjadi bila **kondisi** bernilai `false` pada saat *do while* mulai dieksekusi.

Sebagai contoh pertama dari skema *do while*, kita akan membuat *method* dengan spesifikasi yang sama seperti *method* kelipatan3Menaik pada skema *while*, namun kali ini kita akan menggunakan skema perulangan *do while*. *Method* ini akan kita beri nama kelipatan3MenaikDoWhile. Kita masih menggunakan *class* DemoPerulangan.

```
public String kelipatan3MenaikDoWhile(int batasBawah, int batasAtas){
    String result="";
    int currentNumber=batasBawah;
    do{
        if(currentNumber%3==0){
            result=result+currentNumber+" ";
        }
        currentNumber++;
    }while(currentNumber<=batasAtas);
    result=result.trim();
    return result;
}
```

Gambar 6. Potongan Kode Program kelipatan3MenaikDoWhile.

21. Apa isi objek *String* yang dikembalikan bila batasBawah>batasAtas dan batasBawah merupakan kelipatan 3? Apakah hasil ini sesuai dengan spesifikasi yang diminta? Jelaskan.
22. Sifat apa dari *do while* yang menyebabkan kesalahan ini terjadi?

Perbaikilah *method* kelipatan3MenaikDoWhile di atas agar dia tidak melakukan kesalahan itu lagi. Anda tetap harus menggunakan *do while*. Contoh di atas berguna untuk mengilustrasikan bahwa tidak semua skema perulangan cocok untuk digunakan di semua kasus. Anda perlu untuk memilih skema perulangan yang tepat. Kumpulkan *project* ini dengan nama M0604xxyyy.jar.

G. *For*

Skema *for* biasanya digunakan untuk **melakukan perulangan yang kita sudah ketahui dengan pasti jumlah perulangan yang akan/perlu dilakukan**. Contohnya adalah membuat String yang berisi perulangan nama anda sebanyak 5 kali; dan menghitung hasil perkalian antara dua buah bilangan dengan cara melakukan penambahan secara berulang-ulang.

Perhatikan sintaks for di bawah ini :

```
for(inisialisasi; kondisi; perubahan)
   statement(s)
```

Keterangan:

- **inisialisasi**: satu atau lebih *statement* yang dieksekusi hanya satu kali pada awal dari perulangan dan sesudah itu tidak pernah dieksekusi lagi. Biasanya diisi dengan inisialisasi variabel pencacah atau *counter*.
- **kondisi**: biasanya berisi ekspresi yang nilainya akan menentukan apakah bagian ***statement*(*s*)** akan dikerjakan atau tidak. Jika ekspresi pada **kondisi** menghasilkan nilai `true`, maka ***statement*(*s*)** akan dieksekusi. Dengan kata lain, program akan masuk ke bagian perulangan dan mengeksekusi semua *statement* yang ada di dalamnya. Jika menghasilkan nilai `false`, maka ***statement*(*s*)** tidak dieksekusi. Hal ini membuat program akan langsung melanjutkan ke *statement* berikutnya setelah skema *for*.
- **perubahan**: setelah semua *statement* pada bagian perulangan dieksekusi, program akan melakukan eksekusi pada bagian perubahan. Yang diubah biasanya nilai dari *counter*. Setelah bagian ini dijalankan, akan dilakukan lagi pengetesan terhadap **kondisi**. Perulangan ini akan berjalan terus sampai saat **kondisi** bernilai `false`.

23. Berapa kalikah jumlah eksekusi minimal ***statement*(*s*)** dari skema *for*? Petunjuk: jumlah eksekusi minimal akan terjadi bila kondisi bernilai `false` pada saat *for* mulai dieksekusi.

Untuk contoh pertama penggunaan *for*, kali ini kita akan membuat sebuah *method* pada *class* DemoPerulangan yang bernama printAngkaMenaik. *Method* ini akan menerima 2 buah *integer*, yaitu batas bawah dan batas atas. *Method* ini akan mengembalikan sebuah objek *string* yang berisi seluruh angka dari batas bawah sampai dengan batas atas secara terurut menaik. Kode dari *method* ini dapat dilihat di bawah ini. Perhatikan komentar-komentar yang ditulis pada *method* tersebut.

```
public String printAngkaMenaik(int batasBawah, int batasAtas){
    String result ="";
    for(int i = batasBawah;i<=batasAtas;i++){
        /**
           * i adalah singkatan dari iterator. Biasanya kita menggunakan nama i,j,k dan seterusnya untuk iterator.
           * Kita menggunakan nama tersebut jika kita tidak memerlukan nama deskriptif untuk kasus yang sedang dibuat.
           * Jika kita memerlukan nama deskriptif, seperti indexX, indexY, pengali dan lain-lain untuk kasus yang sedang dibuat,
           * maka akan lebih baik jika menggunakan nama deskriptif tersebut.
        */

        result=result+i+" ";
    }
    result = result.trim();
    return result;
}
```

Gambar 7. Potongan Kode Program printAngkaMenaik.

24. Apakah *blok statement* milik *for* pada *method* di atas akan dijalankan bila batasAtas lebih kecil dari batasBawah?

Tambahkanlah sebuah *method* baru dengan nama printAngkaMenurun yang akan mengeluarkan seluruh angka dari batas atas sampai dengan batas bawah secara terurut menurun. Sama seperti *method* printAngkaMenaik, *method* ini juga akan menggunakan skema perulangan *for*. Kumpulkan *project* ini dengan nama M0605xxyyy.jar.

Tugas *While*

Tugas ini fokus dalam melatih kemampuan Anda dalam menggunakan *while* dalam sebuah kasus. Oleh karena itu, desain *class* yang diberikan tidaklah menjadi pertimbangan utama dalam perancangan soal ini. Buatlah sebuah *class* dengan nama TugasWhile. *Class* ini akan memiliki sebuah *method* berikut ini : `public int pangkat(int x,int y)`. *Method* ini akan mengembalikan hasil x^y, dengan x dan y adalah *integer* yang lebih besar atau sama dengan 0. Pastikan bahwa *method* Anda juga berjalan dengan benar bila y bernilai 0 ($x^0 = 1$). Gunakan skema pengulangan *while*. Anda **tidak boleh** menggunakan fungsi `Math.pow` dari Java. Ide pengerjaan dari soal ini adalah bahwa nilai x pangkat y dapat dicari dengan cara melakukan perkalian secara berulang-ulang. Sebagai contoh, $2^3 = 2 \times 2 \times 2$. Kumpulkan berkas jawaban dengan nama M0606xxyyy.jar

Tugas *For*

Tugas ini akan fokus pada melatih kemampuan anda dalam menggunakan *for* dalam sebuah kasus. Oleh karena itu, desain *class* yang diberikan tidaklah menjadi pertimbangan utama dalam perancangan soal ini. Buatlah sebuah *class* dengan nama TugasFor. *Class* ini akan memiliki sebuah *method* berikut ini : `public int kali(int operand1,int operand2)`. *Method* ini akan mengembalikan hasil dari operand1 × operand2 dengan cara melakukan penambahan secara berulang-ulang. Operand1 dan operand2 adalah *integer*>=0. Gunakanlah skema perulangan *for*. Anda **tidak boleh** menggunakan operator perkalian "*". Ide untuk mengerjakan method ini adalah bahwa operasi perkalian operand1 × operand2 dapat diubah menjadi operasi penjumlahan sebanyak operand2 kali. Sebagai contoh, $4 \times 6 = 4 + 4 + 4 + 4 + 4 + 4$. Kumpulkan berkas jawaban dengan nama M0607xxyyy.jar