

PEMROGRAMAN BERORIENTASI OBYEK

Dosen Pengampu : Bpk. Andi Iwan Nurhidayat, S.Kom., M.T.



DISUSUN OLEH :

MUHAMMAD LADZI SAFRONI 19051397074

D4 Manajemen Informatika B

**UNIVERSITAS NEGERI SURABAYA
UNESA**

JURUSAN TEKNIK INFORMATIKA

Jl. Ketintang, Ketintang, Kec. Gayungan, Kota Surabaya, Jawa Timur 60231, Telp. (031) 8280009

SOAL

1. Seorang analyst membuat aplikasi simulasi permainan. Analyst membuat class diagram dengan abstract class sebagai berikut :

<i>Permainan</i>
-namaPemain : string -levelPemain : int
+setNamaPemain(in namaPemain : string) : void +setLevelPemain(in levelPemain : int) : void +getNamaPemain() : string +getLevelPemain() : int +jalankan() : void <i>+hitungSkor(in hit : int, in miss : int) : int</i>

Deskripsi:

- Atribut namaPemain dan levelPemain menyimpan nama dan level pemain.
- Nilai levelPemain berkisar 1-100 dengan ketentuan :
 - 1-20: normal
 - 21-80: medium
 - 81-100: hard
- Method jalankan() akan menjalankan skenario permainan (set nama dan level pemain, mengeluarkan data tersebut, dan menghitung skor pemain).
- Method hitungSkor merupakan abstract method.

Tugas:

- Buatlah kode berdasarkan abstract class di atas.
- Buatlah 2 classes lain yang menggunakan abstract class di atas dengan deskripsi :
 - "PermainanArcade" dengan aturan hitung skor: jumlah hit x 3 – jumlah miss x 1;
 - "PermainanStrategy" dengan aturan hitung skor: jumlah hit x 5; Perhatikan bahwa algoritma hitungSkor ditentukan oleh subclass, bukan superclass.
 - Buktikan bahwa abstract method memastikan bahwa method tersebut di-override oleh subclass.
 - Buktikan bahwa objek dari abstract class tidak dapat dibentuk.

JAWABAN

a) Buatlah kode berdasarkan abstract class di atas.

Jawab :

```
package simulasi;

abstract class Permainan {
    String namaPemain;
    int levelPemain;
    private int hitungSkor;

    public void setNamaPemain(String namaPemain)
    {
        this.namaPemain = namaPemain;
    }

    public void setLevelPemain(int levelPemain)
    {
        this.levelPemain = levelPemain;
    }

    public String getNamaPemain()
    {
        return namaPemain;
    }

    public int getLevelPemain()
    {
        return levelPemain;
    }

    public void jalankan()
    {
        setNamaPemain("Muhammad Ladzi Safroni");
        setLevelPemain(80);
        System.out.println("Nama Pemain      : "+getNamaPemain());
        System.out.print("Level Pemain      : "+getLevelPemain()+" ");
        if (getLevelPemain() >= 1 && getLevelPemain() <=20 )
        {
            System.out.println("Normal");
        }
        else if (getLevelPemain() >=21 && getLevelPemain() <=80)
        {
            System.out.println("Medium");
        }
    }
}
```

```

        }
        else if (getLevelPemain() >= 81 &&
getLevelPemain() <= 100)
        {
            System.out.println("Hard");
        }
        System.out.println("Skor Permainan :
"+hitungSkor(5,3));
    }

    public abstract int hitungSkor(int hit, int miss);
}

class PermainanArcade extends Permainan {

    @Override
    public int hitungSkor(int hit, int miss) {
        return hit*3-miss*1;
    }
}

class PermainanStrategy extends Permainan {

    @Override
    public int hitungSkor(int hit, int miss) {
        return hit*5;
    }
}

public class SimulasiPermainan {
    public static void main(String args[]){
        System.out.println("Aplikasi Simulasi
Permainan");
        System.out.println();
        System.out.println("Permainan Arcade : ");
        Permainan arcade = new PermainanArcade();
        arcade.jalankan();
        System.out.println();
        System.out.println("Permainan Strategy : ");
        Permainan strategy = new PermainanStrategy();
        strategy.jalankan();
        System.out.println();
        System.out.println("Program By : Muhammad
Ladzi Safroni | 074 | D4 MI 2019B");
        System.out.println();
    }
}

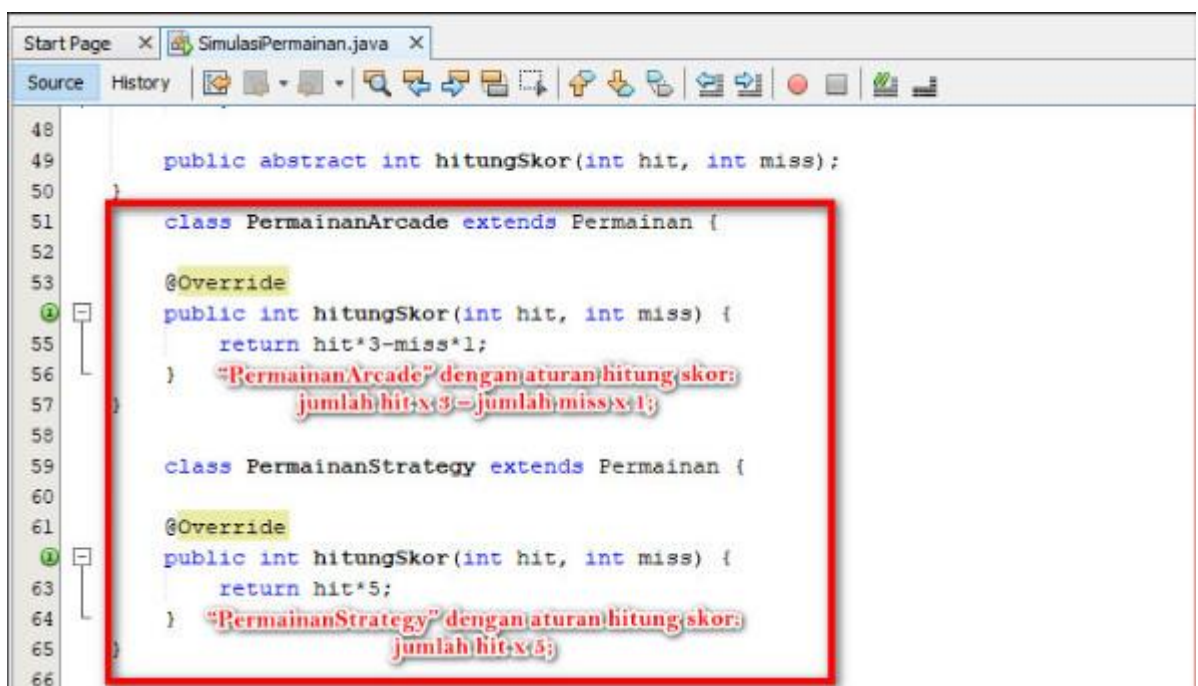
```

b) Buatlah 2 classes lain yang menggunakan abstract class di atas dengan deskripsi :

- “PermainanArcade” dengan aturan hitung skor: jumlah hit x 3 – jumlah miss x 1;
- “PermainanStrategy” dengan aturan hitung skor: jumlah hit x 5; Perhatikan bahwa algoritma hitungSkor ditentukan oleh subclass, bukan superclass.
- Buktikan bahwa abstract method memastikan bahwa method tersebut di-override oleh subclass.
- Buktikan bahwa objek dari abstract class tidak dapat dibentuk.

Jawab :

- “PermainanArcade” dengan aturan hitung skor: jumlah hit x 3 – jumlah miss x 1;
- “PermainanStrategy” dengan aturan hitung skor: jumlah hit x 5; Perhatikan bahwa algoritma hitungSkor ditentukan oleh subclass, bukan superclass.



```
48
49     public abstract int hitungSkor(int hit, int miss);
50 }
51
52 class PermainanArcade extends Permainan {
53     @Override
54     public int hitungSkor(int hit, int miss) {
55         return hit*3-miss*1;
56     }
57     "PermainanArcade" dengan aturan hitung skor:
58     jumlah hit x 3 - jumlah miss x 1;
59
60 class PermainanStrategy extends Permainan {
61     @Override
62     public int hitungSkor(int hit, int miss) {
63         return hit*5;
64     }
65     "PermainanStrategy" dengan aturan hitung skor:
66     jumlah hit x 5;
```

- c. Buktikan bahwa abstract method memastikan bahwa method tersebut di-override oleh subclass.

```
48 public abstract int hitungSkor(int hit, int miss);
49
50 }
51
52 class PermainanArcade extends Permainan {
53     @Override
54     public int hitungSkor(int hit, int miss) {
55         return hit*3-miss*1;
56     }
57 }
58
59 class PermainanStrategy extends Permainan {
60     @Override
61     public int hitungSkor(int hit, int miss) {
62         return hit*5;
63     }
64 }
65
66 }
```

Abstract Method

di-override oleh subclass PermainanArcade

di-override oleh subclass PermainanStrategy

- d. Buktikan bahwa objek dari abstract class tidak dapat dibentuk.

```
3 abstract class Permainan {
4     String namaPemain;
5     int levelPemain;
6     private int hitungSkor;
7
8     public void setNamaPemain(String namaPemain)
9     {
10         this.namaPemain = namaPemain;
11     }
12
13     public void setLevelPemain(int levelPemain)
14     {
15         this.levelPemain = levelPemain;
16     }
17
18     public String getNamaPemain()
19     {
20         return namaPemain;
21     }
22
23     public int getLevelPemain()
24     {
25         return levelPemain;
26     }
27
28     public abstract int hitungSkor(int hit, int miss);
29 }
```

objek dari abstract class tidak dapat dibentuk

objek dari abstract class tidak dapat dibentuk

objek dari abstract class tidak dapat dibentuk