**LAPORAN**

**PRAKTIKUM PEMROGRAMAN**

**ABSTRACT CLASS DAN INTERFACES**

****

**Oleh:**

Muchamad Lutfi Maftuh

NIM. 19537141023

**PROGRAM STUDI TEKNOLOGI INFORMASI**

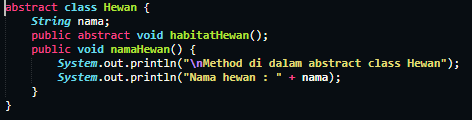
**JURUSAN PENDIDIKAN TEKNIK ELEKTRONIKA DAN INFORMATIKA**

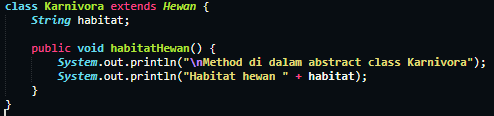
**FAKULTAS TEKNIK**

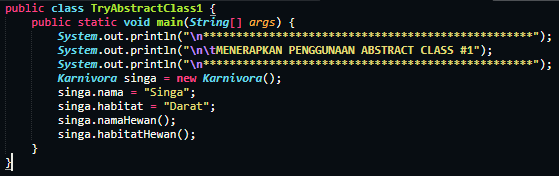
**UNIVERSITAS NEGERI YOGYAKARTA**

**2020**

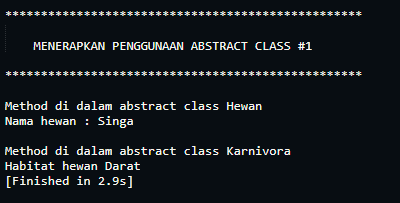
1. **LATIHAN**
2. Menggunakan abstract class. Terdapat ***class Hewan, class Karnivora, dan TryAbstractClass1.***







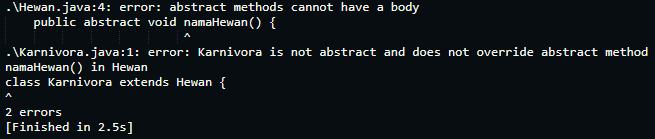
1. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



1. Berikan penjelasan terkait jalannya program ini!

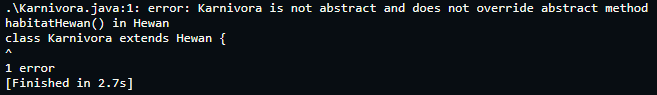
* **Class Hewan** merupakan kelas abstract yang berisi data field String nama, method abstract void habitatHewan(), dan method void namaHewan().
* **Class Karnivora** meng-extends Hewan yang berarti harus mengoverride method abstract void habitatHewan(). Karnivora memiliki data field habitat.
* **Class TryAbstractClass1** membuat object Karnivora singa. Kemudian menginisiasi dengan perintah singa.nama (di Class Hewan sesuai polymorphism) dan singa.habitat (di Class Karnivora). Method namaHewan() dan habitatHewan pun dipanggil untuk menampilkan kalimat seperti pada gambar point a.

1. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika method namaHewan() diubah menjadi method abstract!



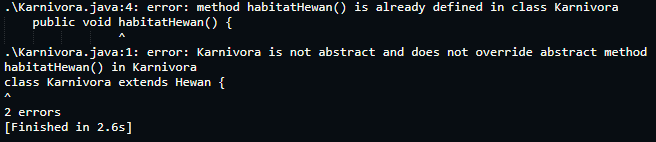
Akan terjadi error yang pertama, method namaHewan tidak boleh memiliki body {} karena abstract. Kedua, Karnivora yang meng-extends Hewan tidak meng-override namaHewan().

1. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika tidak dilakukan overriding terhadap abstract method habitatHewan()!



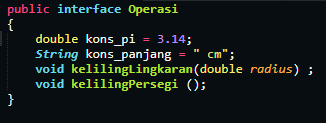
Terjadi error karena method abstract harus di-override oleh siapapun yang meng-extends-nya

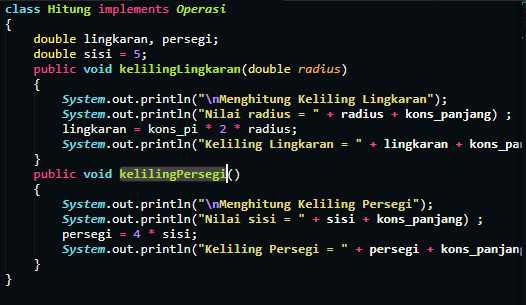
1. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika abstract method habitatHewan() dideklarasikan dalam class Karnivora!

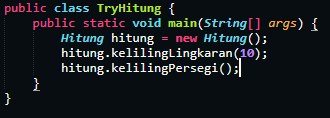


Terjadi error karena terdapat dua method yang sama namanya. Juga error karena Karnivora bukan abstract dan abstract method harus di-override.

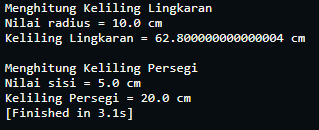
1. Menggunakan interface. Terdapat ***interface Operasi, class Hitung, class TryHitung (Buat sendiri).***







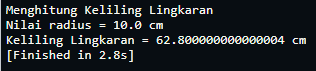
1. Buatlah sebuah class baru yang berisi method main untuk menjalankan programtersebut! Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



1. Berikan penjelasan terkait jalannya program ini!

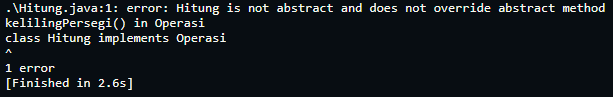
* **Interface Operasi** memiliki data fields konst\_pi dan kons\_panjang. Juga memiliki method abstract kelilingLingkaran(double radius) dan kelilingPersegi().
* **Class Hitung** meng-implements Operasi yang berarti harus meng-override method abstract diatas.
* **Class TryHitung** mencoba membuat Object Hitung dan memanggil method yang sudah di-override.

1. Tunjukkan hasil kompilasi dan eksekusi program kemudian berikan penjelasan singkat jika method kelilingPersegi() dikosongkan!



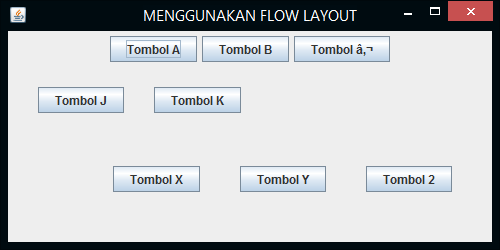
Method kelilingPersegi tetap dipanggil tetapi tidak melakukan apa-apa karena kosong.

1. Tunjukkan hasil kompilasi program dan berikan penjelasan singkat jika method kelilingPersegi()dihapus dan tidak dipanggil dalam method main!



Terjadi error dikarenakan method abstract kelilingPersegi() yang harusnya di-override oleh Hitung, tidak di-override.

1. Menggunakan class FlowLayout.
2. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



1. Berikan penjelasan terkait jalannya program ini!

Mencoba menggunakan FlowLayout yang akan diset ke sebuah JPanel.

1. Jelaskan fungsi perintah

* p2.setLayout(new FlowLayout (FlowLayout.LEFT,30,20));

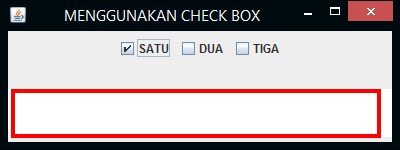
Perintah diatas digunakan untuk meng-set layout pada p2 (JPanel) dengan FlowLayout berparameter:

* **FlowLayout.LEFT** : align left.
* **30** : HorizontalGap
* **20** : VerticalGap
* p3.setLayout(new FlowLayout (FlowLayout.RIGHT,40,50));

Perintah diatas digunakan untuk meng-set layout pada p2 (JPanel) dengan FlowLayout berparameter:

* **FlowLayout.RIGHT** : align right.
* **40** : HorizontalGap
* **50** : VerticalGap

1. Menggunakan class JCheckBox.
2. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



1. Berikan penjelasan terkait jalannya program ini!

Program tersebut mencoba membuat checkbox dengan JCheckBox. Ketika terjadi perubahan pada checkbox **harusnya** muncul text pada textArea.

1. Jelaskan fungsi perintah

* tArea = new JTextArea(3,20);

Perintah diatas digunakan menginisiasi object tArea dengan constructor JTextArea. Angka 3 adalah tingginya dan angka 20 adalah panjangnya.

* tArea.setEditable(false);

Perintah diatas digunakan untuk meng-set agar tArea tidak bisa diedit.

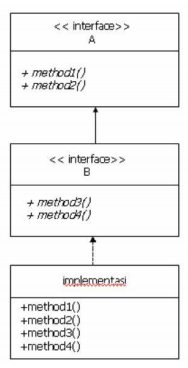
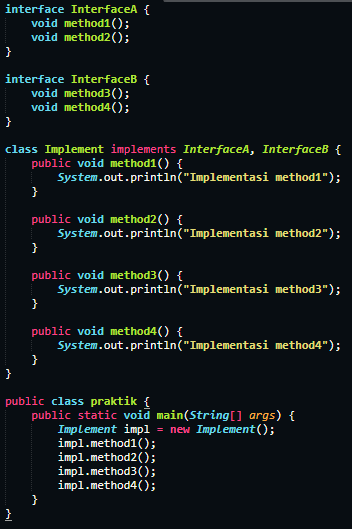
1. Menggunakan class JRadioButton.
2. Lakukan kompilasi dan eksekusi program kemudian tunjukkan hasilnya!



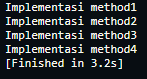
1. Berikan penjelasan terkait jalannya program ini!

Program diatas menggunakan JRadioButton dan ketika terjadi ActionEvent textarea akan menampilkan pilihan.

1. **TUGAS PRAKTIKUM**
2. Impementasikan UML class diagram dibawah!

Hasil run:



1. **TUGAS RUMAH**
2. Berikan argumentasi anda tentang perbedaan antara Interface dan Abstract? Sertakan contoh program untuk memperkuat argumen anda!

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| Bisa berisi abstract dan non-abstract method. | Hanya boleh berisi abstract method. |
| Modifiersnya harus dituliskan sendiri. | Tidak perlu menulis public abstract di depan nama method. Karena secara implisit, modifier untuk method di interface adalah **public** dan ***abstract***. |
| Bisa mendeklarasikan **constant** dan **instance variable**. | Hanya bisa mendeklarasikan **constant**.  Secara implisit variable yang dideklarasikan di interface bersifat **public**, **static** dan **final**. |
| Method boleh bersifat **static**. | Method tidak boleh bersifat **static** |
| Met hod boleh bersifat **final**. | Method tidak boleh bersifat **final**. |
| Suatu abstact class hanya bisa meng-*extend* satu abstract class lainnya. | Suatu interface bisa meng-*extend*  satu atau lebih interface lainnya. |
| Suatu abstract class hanya bisa meng-*extend* satu abstract class dan meng-**implement** beberapa interface. | Suatu interface hanya bisa meng-*extend* interface lainnya.  Dan tidak bisa meng-*implement* class atau interface lainnya. |

1. Jelaskan kondisi yang tepat untuk penggunaan Abstract dan Interface!

**Abstract** untuk mendefinisikan secara luas sifat-sifat dari class tertinggi pada hirarki OOP, dan gunakan subclassnya (turunannya/child class) untuk melengkapi deskripsi method dari class abstract.

**Interface** untuk mendefinisikan method standar yang sama dalam class-class yang berbeda. Sekali kita telah membuat kumpulan definisi method standar(abstrak), kita dapat menulis method tunggal(spesifik) untuk memanipulasi semua class-class yang mengimplementasikan interface tsb.

1. Berikan capaian pemahaman anda dalam bentuk persentase (0%-100%) tentang praktikum pertemuan ini! Tambahkan argumentasi singkat mengenai teknik pembelajaran yang telah dilaksanakan selama praktikum!

**Pemahaman 90%.**