Modul 11. Abstraction

Tugas Pendahuluan :

1. Apa yang dimaksud dengan abstraksi dalam pemrograman dan mengapa itu penting?
2. Apa perbedaan antara class abstrak dan metode abstrak dalam Java?
3. Bagaimana cara mendefinisikan sebuah class abstrak dalam Java? Berikan contoh.
4. Apa yang harus dilakukan oleh sebuah subclass ketika mewarisi sebuah class abstrak yang memiliki metode abstrak?
5. Apa yang terjadi jika sebuah subclass tidak mengimplementasikan semua metode abstrak dari superclassnya?
6. Apa yang dimaksud dengan metode abstrak dalam Java dan bagaimana cara mendefinisikannya?

Materi :

Abstraksi adalah salah satu konsep penting dalam pemrograman berorientasi objek (OOP). Ini mengacu pada kemampuan untuk menyembunyikan rincian-rincian internal suatu objek dan hanya mengekspos informasi yang relevan atau yang penting. Dalam Java, abstraksi dapat dicapai melalui penggunaan class abstrak dan metode abstrak. Berikut adalah beberapa poin penting terkait dengan abstraksi dalam Java:

1. Class Abstrak: Sebuah class abstrak adalah class yang tidak dapat diinstansiasi (tidak dapat membuat objek dari class tersebut) dan biasanya berisi metode-metode abstrak. Class abstrak dideklarasikan dengan kata kunci abstract.
2. Metode Abstrak: Metode abstrak adalah metode yang hanya memiliki deklarasi tanpa implementasi. Mereka dideklarasikan dalam class abstrak dan harus diimplementasikan oleh semua subkelas yang mengextend class tersebut.
3. Tujuan Abstraksi: Tujuan utama abstraksi adalah untuk menyembunyikan kompleksitas dan rincian yang tidak perlu dari pemakai class (pengguna objek). Ini memungkinkan untuk fokus pada apa yang objek dapat lakukan daripada bagaimana mereka melakukannya.
4. Pewarisan dan Polymorphism: Class abstrak sering digunakan sebagai class dasar dalam hierarki pewarisan (inheritance). Subkelas yang mengextend class abstrak akan mengimplementasikan metode-metode abstraknya, dan ini memungkinkan polymorphism.
5. Contoh Praktis: Salah satu contoh penerapan abstraksi adalah dalam pemrograman GUI (Graphical User Interface). Pengguna hanya berinteraksi dengan elemen-elemen antarmuka seperti tombol atau kotak teks tanpa perlu tahu rincian kompleks di baliknya.

6. Kelas Konkrit: Kelas konkrit adalah kelas yang dapat diinstansiasi dan biasanya memiliki implementasi untuk metode-metodenya.
7. Abstraksi dalam Interface: Selain class abstrak, abstraksi juga dapat dicapai melalui penggunaan interface. Interface adalah kontrak yang menggambarkan metode-metode tanpa implementasi.

Abstraksi memungkinkan untuk menciptakan kode yang lebih modular, mudah dipahami, dan mudah diubah. Hal ini juga memungkinkan untuk mengimplementasikan polimorfisme dengan lebih baik, sehingga kode dapat lebih fleksibel dan dapat digunakan kembali. Dalam Java, class abstrak dan interface adalah alat penting yang digunakan untuk mencapai abstraksi. Mereka membantu mengorganisir kode dan menyediakan cara untuk mengabstraksi perilaku objek dalam program Kita.

Abstraksi adalah salah satu konsep fundamental dalam pemrograman berorientasi objek (OOP) yang sangat penting karena membantu dalam:

1. Pemahaman yang Lebih Baik: Abstraksi memungkinkan kita untuk mengisolasi detail yang tidak penting dan fokus pada aspek-aspek penting dari suatu objek atau konsep. Ini membuat kode lebih mudah dipahami, dikelola, dan diterapkan.
2. Penyederhanaan: Dengan menyembunyikan detail yang tidak perlu, abstraksi membuat kode lebih sederhana. Ini membuat pemrograman lebih efisien dan memungkinkan untuk fokus pada solusi masalah yang sebenarnya.
3. Penggunaan Ulang: Abstraksi memungkinkan kita untuk membuat class abstrak dan metode abstrak yang dapat digunakan kembali dalam berbagai konteks. Ini mengurangi pengulangan kode dan meningkatkan efisiensi pengembangan perangkat lunak.
4. Ketergantungan yang Lebih Rendah: Ketika kita mengabstraksi objek atau konsep, kita dapat mengurangi ketergantungan antara berbagai bagian kode. Ini membuat kode lebih fleksibel dan mudah untuk diubah atau diperbaiki tanpa memengaruhi seluruh program.
5. Model Dunia Nyata: Abstraksi memungkinkan kita untuk memodelkan objek-objek dunia nyata dalam program. Misalnya, kita dapat mengabstraksi mobil sebagai objek dengan properti seperti merek, model, dan metode seperti berkendara. Ini membuat pemrograman lebih intuitif dan memungkinkan kita untuk memodelkan solusi yang lebih baik.

6. Keamanan: Abstraksi dapat membantu dalam mengimplementasikan tingkat aksesibilitas (seperti public, private, dan protected) yang dapat meningkatkan keamanan kode. Kita dapat mengizinkan akses hanya ke detail yang dibutuhkan dan menyembunyikan yang lain.
7. Perbaikan Kode: Saat ada kesalahan atau perubahan yang diperlukan, abstraksi memungkinkan kita untuk mengidentifikasi dan memperbaiki masalah lebih cepat. Kita dapat fokus pada komponen yang relevan tanpa harus memeriksa keseluruhan kode.
8. Kepelitian: Abstraksi memungkinkan pengembang untuk bekerja pada bagian-bagian yang berbeda dari sistem secara independen. Ini memungkinkan untuk meningkatkan produktivitas dan mempercepat pengembangan.

Keseluruhan, abstraksi adalah alat penting dalam pemrograman berorientasi objek yang membantu dalam mengorganisasi, menyederhanakan, dan meningkatkan kualitas kode. Hal ini memungkinkan untuk merancang solusi perangkat lunak yang lebih baik, lebih mudah dipahami, dan lebih mudah diatur.

Class Abstrak

Class Abstrak dalam Java adalah class yang tidak dapat diinstansiasi (tidak dapat membuat objek dari class tersebut) dan biasanya berisi metode-metode abstrak. Class abstrak dideklarasikan dengan kata kunci abstract. Dalam class abstrak, Kita dapat mendefinisikan metode-metode abstrak (metode yang hanya memiliki deklarasi tanpa implementasi) yang akan diimplementasikan oleh subkelasnya. Berikut adalah beberapa poin penting tentang class abstrak:

1. Tidak Dapat Diinstansiasi: Class abstrak tidak dapat digunakan untuk membuat objek langsung. Kita hanya dapat membuat objek dari subkelasnya.
2. Metode Abstrak: Class abstrak seringkali memiliki satu atau lebih metode abstrak yang tidak memiliki implementasi. Metode-metode ini hanya memiliki deklarasi, dan subkelas yang meng-extend class abstrak wajib mengimplementasikan metode-metode tersebut.
3. Metode Konkrit: Class abstrak juga dapat memiliki metode dengan implementasi yang lengkap (metode konkrit). Subkelas dapat mewarisi metode konkrit ini atau menggantinya sesuai kebutuhan.
4. Pewarisan: Class abstrak biasanya digunakan sebagai class dasar dalam hierarki pewarisan. Subkelas yang meng-extend class abstrak akan mengimplementasikan metode-metode abstraknya. Ini memungkinkan polimorfisme, di mana objek dari subkelas dapat digunakan dalam konteks class abstrak.

5. Tujuan Abstraksi: Tujuan utama dari class abstrak adalah untuk memungkinkan abstraksi, yaitu menyembunyikan rincian yang tidak perlu dari pemakai class. Ini memungkinkan pemakai class untuk fokus pada apa yang objek dapat lakukan daripada bagaimana mereka melakukannya.

Berikut adalah contoh class abstrak dalam Java:

```
1. // Class abstrak Hewan
 2. abstract class Hewan {
 3.     // Metode abstrak suara yang akan diimplementasikan oleh subkelas
 4.     abstract void suara();
 5.
 6.     // Metode konkrit
 7.     void tidur() {
 8.         System.out.println("Hewan tidur.");
 9.     }
10. }
11.
12. // Subkelas Anjing yang meng-extend class abstrak Hewan
13. class Anjing extends Hewan {
14.     // Implementasi metode abstrak suara
15.     void suara() {
16.         System.out.println("Anjing menggonggong.");
17.     }
18. }
19.
20. // Subkelas Kucing yang meng-extend class abstrak Hewan
21. class Kucing extends Hewan {
22.     // Implementasi metode abstrak suara
23.     void suara() {
24.         System.out.println("Kucing meong.");
25.     }
26. }
27.
28. public class Main {
29.     public static void main(String[] args) {
30.         // Membuat objek dari subkelas Anjing
31.         Hewan anjing = new Anjing();
32.         anjing.suara();
33.         anjing.tidur(); // Memanggil metode konkrit dari class abstrak
34.
35.         // Membuat objek dari subkelas Kucing
36.         Hewan kucing = new Kucing();
```

```
        kucing.suara();
        kucing.tidur(); // Memanggil metode konkrit dari class abstrak
    }
}

```

Dalam contoh di atas, Hewan adalah class abstrak dengan metode abstrak suara(). Subkelas Anjing dan Kucing meng-extend Hewan dan mengimplementasikan metode abstrak suara(). Kita tidak dapat membuat objek langsung dari Hewan, tetapi Kita dapat membuat objek dari subkelasnya dan mengakses metode-metodenya melalui class abstrak.

Inheriting Class Abstract

Dalam Java, ketika Kita mewarisi sebuah class abstrak (class yang mengandung satu atau lebih metode abstrak), Kita memiliki dua pilihan:

1. Implementasi Metode Abstrak: Kita dapat mengimplementasikan semua metode abstrak yang didefinisikan dalam class abstrak tersebut. Ini adalah keharusan; jika tidak, subclass itu sendiri harus dinyatakan sebagai class abstrak.
2. Mendeklarasikan Subclass sebagai Abstrak: Jika Kita tidak ingin mengimplementasikan semua metode abstrak dalam subclass, Kita dapat mendeklarasikan subclass itu sendiri sebagai class abstrak. Ini berarti bahwa setiap subclass konkret yang meng-extend subclass abstrak Kita harus mengimplementasikan metode abstrak yang tersisa.

Berikut adalah contoh untuk mengilustrasikan kedua pilihan tersebut:

```
// Class abstrak
abstract class Bentuk {
    // Metode abstrak
    abstract double luas();

    // Metode konkret
    void tampilkanLuas() {
        System.out.println("Luas: " + luas());
    }
}

// Subclass konkret yang mengimplementasikan metode abstrak
class Lingkaran extends Bentuk {
```

```
    private double radius;

    Lingkaran(double r) {
        radius = r;
    }

    // Implementasi dari metode abstrak
    double luas() {
        return Math.PI * radius * radius;
    }
}

// Subclass abstrak
abstract class Segitiga extends Bentuk {
    // Tidak ada implementasi metode abstrak di sini
}

// Subclass konkret yang meng-extend subclass abstrak dan mengimplementasikan metode abstrak
class SegitigaSamaSisi extends Segitiga {
    private double sisi;

    SegitigaSamaSisi(double s) {
        sisi = s;
    }

    // Implementasi dari metode abstrak
    double luas() {
        return (Math.sqrt(3) / 4) * sisi * sisi;
    }
}

public class Main {
    public static void main(String[] args) {
        Lingkaran lingkaran = new Lingkaran(5);
        lingkaran.tampilkanLuas();

        SegitigaSamaSisi segitiga = new SegitigaSamaSisi(4);
        segitiga.tampilkanLuas();
    }
}

```

Dalam contoh ini:

- Class Bentuk adalah class abstrak dengan metode abstrak luas() dan metode konkret tampilkanLuas().
- Class Lingkaran adalah subclass konkret yang meng-extend Bentuk dan mengimplementasikan metode luas().
- Class Segitiga adalah subclass abstrak dari Bentuk yang tidak mengimplementasikan metode luas().
- Class SegitigaSamaSisi adalah subclass konkret yang meng-extend Segitiga dan mengimplementasikan metode luas().

Kita dapat melihat bahwa Kita memiliki opsi untuk mengimplementasikan metode abstrak dalam subclass konkret (seperti dalam class Lingkaran) atau mendeklarasikan subclass itu sendiri sebagai class abstrak (seperti dalam class Segitiga) untuk menunda implementasi ke subclass konkretnya.

Perbedaan utama antara class abstrak dan class biasa (non-abstrak) dalam Java adalah sebagai berikut:

1. Instansiasi Objek:

Class Biasa: Kita dapat membuat objek dari class biasa (non-abstrak) dan menggunakannya secara langsung. Contoh:

```
1. class Kendaraan {
2.     void berkendara() {
3.         System.out.println("Kendaraan sedang berjalan.");
4.     }
5. }
6.
7. public class Main {
8.     public static void main(String[] args) {
9.         Kendaraan mobil = new Kendaraan();
10.        mobil.berkendara();
11.    }
12. }
13.
```

Class Abstrak: Kita tidak dapat membuat objek langsung dari class abstrak. Class abstrak digunakan sebagai blueprint untuk class-class turunannya (subclass). Contoh:

```
1. abstract class Kendaraan {
2.     abstract void berkendara();
3. }
4.
5. class Mobil extends Kendaraan {
6.     void berkendara() {
7.         System.out.println("Mobil sedang berjalan.");
8.     }
9. }
10.
```

```
11. public class Main {
12.     public static void main(String[] args) {
13.         Kendaraan mobil = new Mobil(); // Membuat objek dari subclass
14.         mobil.berkendara();
15.     }
16. }
17.
```

2. Metode Abstrak:

Class Biasa: Class biasa dapat memiliki metode biasa (non-abstrak) yang memiliki implementasi (kode) yang lengkap. Kita dapat memanggil metode ini langsung dari objeknya.

Class Abstrak: Class abstrak dapat memiliki metode abstrak yang tidak memiliki implementasi dalam class tersebut. Metode abstrak hanya memiliki deklarasi tanpa kode. Subclass yang mewarisi class abstrak harus mengimplementasikan (mengisi kode) untuk metode abstrak tersebut.

3. Tujuan:

Class Biasa: Class biasa digunakan ketika Kita ingin membuat objek-objek yang dapat digunakan secara langsung. Mereka biasanya memiliki implementasi lengkap untuk metode-metodenya.

Class Abstrak: Class abstrak digunakan ketika Kita ingin menciptakan sebuah blueprint atau kerangka dasar untuk class-class turunannya. Mereka sering digunakan untuk memodelkan konsep umum yang memerlukan implementasi khusus dalam subclass.

4. Contoh Penggunaan:

Class Biasa:

```
 1. class Hewan {
 2.     void bersuara() {
 3.         System.out.println("Hewan ini bersuara.");
 4.     }
 5. }
 6.
 7. class Kucing extends Hewan {
 8.     void bersuara() {
 9.         System.out.println("Kucing mengeluarkan suara meow.");
10.     }
11. }
12.
13. public class Main {
14.     public static void main(String[] args) {
15.         Hewan hewan = new Kucing();
16.         hewan.bersuara(); // Output: Kucing mengeluarkan suara meow.
17.     }
18. }
19.
```

Class Abstrak:

```
abstract class Bentuk {
    abstract void hitungLuas();
}

class Persegi extends Bentuk {
    void hitungLuas() {
        System.out.println("Menghitung luas persegi...");
    }
}

public class Main {
    public static void main(String[] args) {
        Bentuk bentuk = new Persegi();
        bentuk.hitungLuas(); // Output: Menghitung luas persegi...
    }
}

```

Dalam contoh di atas, class Hewan adalah class biasa dengan metode yang memiliki implementasi lengkap. Sedangkan, class Bentuk adalah class abstrak dengan metode abstrak yang harus diimplementasikan oleh subclass seperti Persegi.

Metode Abstrak

Metode Abstrak adalah metode dalam sebuah class yang hanya memiliki deklarasi, tetapi tidak memiliki implementasi. Metode ini ditkitai dengan kata kunci abstract. Metode abstrak digunakan untuk mendefinisikan "kerangka" metode yang harus diimplementasikan oleh subkelasnya. Dalam class yang mengandung metode abstrak, Kita tidak memberikan implementasi sebenarnya; sebaliknya, Kita hanya memberikan tkita arahan tentang apa yang metode tersebut harus lakukan. Berikut adalah beberapa poin penting tentang metode abstrak:

1. Tidak Ada Implementasi: Metode abstrak tidak memiliki blok kode yang sebenarnya. Mereka hanya memiliki deklarasi tanpa tubuh (body) metode.
2. Subkelas Wajib Implementasi: Subkelas yang meng-extend class yang memiliki metode abstrak harus mengimplementasikan metode abstrak tersebut. Ini adalah kontrak yang memastikan bahwa subkelas memiliki implementasi untuk metode tersebut.
3. Konteks Polimorfisme: Metode abstrak digunakan dalam konteks polimorfisme. Ini memungkinkan objek dari subkelas untuk digunakan secara umum dalam konteks class yang memiliki metode abstrak.
4. Abstraksi: Tujuan utama metode abstrak adalah untuk menyediakan abstraksi atau pemodelan tingkat tinggi dari suatu perilaku yang harus dimiliki oleh objek dalam class tersebut.

Berikut adalah contoh metode abstrak dalam Java:

```
// Class abstrak Hewan
abstract class Hewan {
    // Metode abstrak suara yang harus diimplementasikan oleh subkelas
    abstract void suara();
}

// Subkelas Anjing yang meng-extend class abstrak Hewan
class Anjing extends Hewan {
    // Implementasi metode abstrak suara
    void suara() {
        System.out.println("Anjing menggonggong.");
    }
}

// Subkelas Kucing yang meng-extend class abstrak Hewan
class Kucing extends Hewan {
    // Implementasi metode abstrak suara
    void suara() {
        System.out.println("Kucing meong.");
    }
}

public class Main {
    public static void main(String[] args) {
        // Membuat objek dari subkelas Anjing
        Hewan anjing = new Anjing();
        anjing.suara(); // Memanggil metode abstrak

        // Membuat objek dari subkelas Kucing
        Hewan kucing = new Kucing();
        kucing.suara(); // Memanggil metode abstrak
    }
}

```

Dalam contoh di atas, Hewan adalah class abstrak dengan metode abstrak suara(). Subkelas Anjing dan Kucing meng-extend Hewan dan mengimplementasikan metode abstrak suara(). Subkelas wajib mengimplementasikan metode abstrak ini atau mereka akan menjadi class abstrak juga.

Latihan :

Latihan 1 : Program berikut mencoba mengimplementasikan konsep abstraksi untuk bentuk geometri. Namun, ada kesalahan dalam implementasi. Perbaiki program ini.

```
abstract class Bentuk {
    abstract double hitungLuas();

    void tampilkanLuas() {
        System.out.println("Luas: " + hitungLuas())
    }
}

class PersegiPanjang extends bentuk {
    private double panjang;
    private double lebar;

    PersegiPanjang(double p, double l) {
        panjang = p;
        lebar = l
    }

    double hitungLuas() {
        return panjang * lebar;
    }
}

public class Main {
    public static void Main(String[] args) {
        Bentuk bentuk = new Bentuk();
        bentuk.tampilkanLuas();
    }
}

```

Instruksi: Perbaiki program di atas agar dapat dijalankan dengan benar dan menghasilkan luas dari sebuah persegi panjang.

Latihan 2: Program ini mencoba menggunakan konsep abstraksi untuk menggambarkan binatang dan suara yang mereka hasilkan. Namun, ada kesalahan dalam implementasi. Perbaiki program ini.

```
abstract class Binatang {
    abstract void suara();

    void tampilkanSuara() {
        System.out.println("Suara: ");
        suara();
    }
}

class Kucing extends Binatang {
    void Suara() {
        System.out.println("Meow!");
    }
```

```
14. }
15.
16. class Anjing extends Binatang {
17.     void suara() {
18.         System.out.println("Woof!")
19.     }
20. }
21.
22. public class Main {
23.     public static void main(String[] args) {
24.         Binatang binatang = new binatang();
25.         binatang.tampilkanSuara()
26.     }
27. }
28.
```

Instruksi: Perbaiki program di atas agar dapat dijalankan dengan benar dan menampilkan suara dari sebuah binatang (kucing atau anjing).

Latihan 3 : Program ini mencoba mengimplementasikan konsep abstraksi untuk berbagai kendaraan. Namun, ada kesalahan dalam implementasi. Perbaiki program ini.

```
 1. abstract class Kendaraan {
 2.     abstract void bergerak();
 3.
 4.     void tampilkanGerakan() {
 5.         System.out.println("Kendaraan bergerak dengan cara: ");
 6.         Bergerak()
 7.     }
 8. }
 9.
10. class Mobil extends Kendaraan {
11.     void bergerak() {
12.         System.out.println("Menggunakan roda.");
13.     }
14. }
15.
16. class Kapal extends Kendaraan {
17.     void bergerak() {
18.         System.out.println("Menggunakan air.");
19.     }
20. }
21.
22. public class Main {
23.     public static void main(String[] args) {
24.         Kendaraan kendaraan = new Kendaraan()
25.         kendaraan.tampilkanGerakan();
26.     }
27. }
28.
```

Instruksi: Perbaiki program di atas agar dapat dijalankan dengan benar dan menampilkan cara bergerak dari sebuah kendaraan (mobil atau kapal).

Latihan 4 : Program berikut mencoba mengimplementasikan konsep abstraksi untuk perangkat elektronik. Namun, ada kesalahan dalam implementasi. Perbaiki program ini.

```
abstract class PerangkatElektronik {
    abstract void Hidupkan();

    void tampilkanStatus() {
        System.out.println("Status: ")
        hidupkan();
    }
}

class Televisi extends PerangkatElektronik {
    void hidupkan() {
        System.out.println("Televisi dinyalakan.");
    }
}

class Laptop extends PerangkatElektronik {
    void hidupkan() {
        System.out.println("Laptop dinyalakan.");
    }
}

public class Main {
    public static void main(String[] args) {
        PerangkatElektronik perangkat = new PerangkatElektronik();
        perangkat.tampilkanStatus()
    }
}

```

Instruksi: Perbaiki program di atas agar dapat dijalankan dengan benar dan menampilkan status dari sebuah perangkat elektronik (televisi atau laptop).

Latihan 5 : Program berikut mencoba menggunakan konsep abstraksi untuk menggambarkan jenis makanan dan cara memasaknya. Namun, ada kesalahan dalam implementasi. Perbaiki program ini.

```
abstract class Makanan {
    abstract void masak();

    void tampilkanCaraMemasak() {
        System.out.println("Cara memasak: ");
        masak()
    }
}

class NasiGoreng extends Makanan {
    void masak() {
        System.out.println("Tumis bawang putih dan bawang merah, masukkan nasi, tambahkan kecap, aduk rata.");
    }
}

class Pizza extends Makanan {
    void Masak() {
        System.out.println("Gulung adonan pizza, tambahkan saus tomat, keju, dan topping lainnya, panggang dalam oven.")
    }
```

```
20. }
21.
22. public class main {
23.     public static void main(String[] args) {
24.         Makanan makanan = new Makanan();
25.         makanan.tampilkanCaraMemasak();
26.     }
27. }
28.
```

Instruksi: Perbaiki program di atas agar dapat dijalankan dengan benar dan menampilkan cara memasak dari sebuah jenis makanan (nasi goreng atau pizza).

Tugas Rumah :

Deskripsi Tugas: Buatlah beberapa class abstrak dan implementasikan metode abstrak di dalamnya. Selain itu, buat beberapa objek dari class tersebut dan memanggil metodenya untuk menghasilkan keluaran yang sesuai.

Langkah-langkah:

1. Buatlah sebuah class abstrak bernama BangunDatar dengan satu metode abstrak hitungLuas(). Class ini akan digunakan sebagai superclass untuk berbagai bentuk geometri.
2. Buatlah dua subclass dari BangunDatar, yaitu Persegi dan Lingkaran. Implementasikan metode hitungLuas() untuk kedua subclass ini sesuai dengan rumus luas persegi dan lingkaran.
3. Buat sebuah class Main yang akan menjadi program utama. Di dalam Main, buat objek dari class Persegi dan Lingkaran, lalu panggil metode hitungLuas() untuk masing-masing objek dan tampilkan hasilnya.
4. Lakukan hal yang sama dengan class abstrak yang berfokus pada bidang lain, seperti Binatang dengan metode abstrak suara(), dan buat beberapa subclass seperti Kucing dan Anjing yang mengimplementasikan metode suara() sesuai dengan suara masing-masing binatang.

Instruksi Tambahan:

1. Pastikan untuk memberikan komentar yang jelas di dalam kode program yang menjelaskan bagaimana konsep abstraksi diterapkan.
2. Uji program Kita dengan membuat objek-objek dari subclass yang sudah dibuat dan memanggil metode-metodenya untuk melihat hasil keluaran yang sesuai.