Prinsip Perancangan Kelas

Brigida Arie Minartiningtyas, M.Kom.

Definisi Kelas

Field (Variabel)

- menyimpan data untuk setiap objek (implementasi dari atribut)

Constructor

- setup objek di awal

Method

- implementasi perilaku objek

Definisi Kelas

```
class ClassName
  {
      Fields
      Constructors
      Methods
  }
```

Petunjuk Pendefinisian Kelas

- Deklarasikan variabel (data member) sebagai private untuk menjamin integritas kelas. Perwujudan enkapsulasi
- Deklarasikan fungsi dengan public untuk menyediakan akses kepada klien kelas. Fungsi public menentukan perilaku objek dari kelas.
- Selalu definisikan suatu konstruktor dan inisialisasikan variabel instance secara lengkap dalam konstruktor sehingga objek akan dibuat dalam suatu kondisi yang valid
- Deklarasikan konstanta sebagai **public** apabila nilainya akan diakses oleh klien atau kelas lain. Jika hanya digunakan secara internal deklarasikan sebagai **private**

UNIFIED
MODELING
LANGUAGE™

Berisi symbol-symbol yang digunakan untuk menggambarkan arsitektur suatu kelas

Behavioral UML Diagram

- Activity Diagram
- Use Case Diagram
- Interaction Overview Diagram
- Timing Diagram
- State Machine Diagram
- Communication Diagram
- Sequence Diagram

Structural UML Diagram

- **Class Diagram**
- Object Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diagram
- Package Diagram
- Profile Diagram

Calendar

+month: String
+day: int
+year: int

+writeOutput(): void
+readInput(): void
+getDay(): int
+getYear(): int
+getMonth(): int

Index

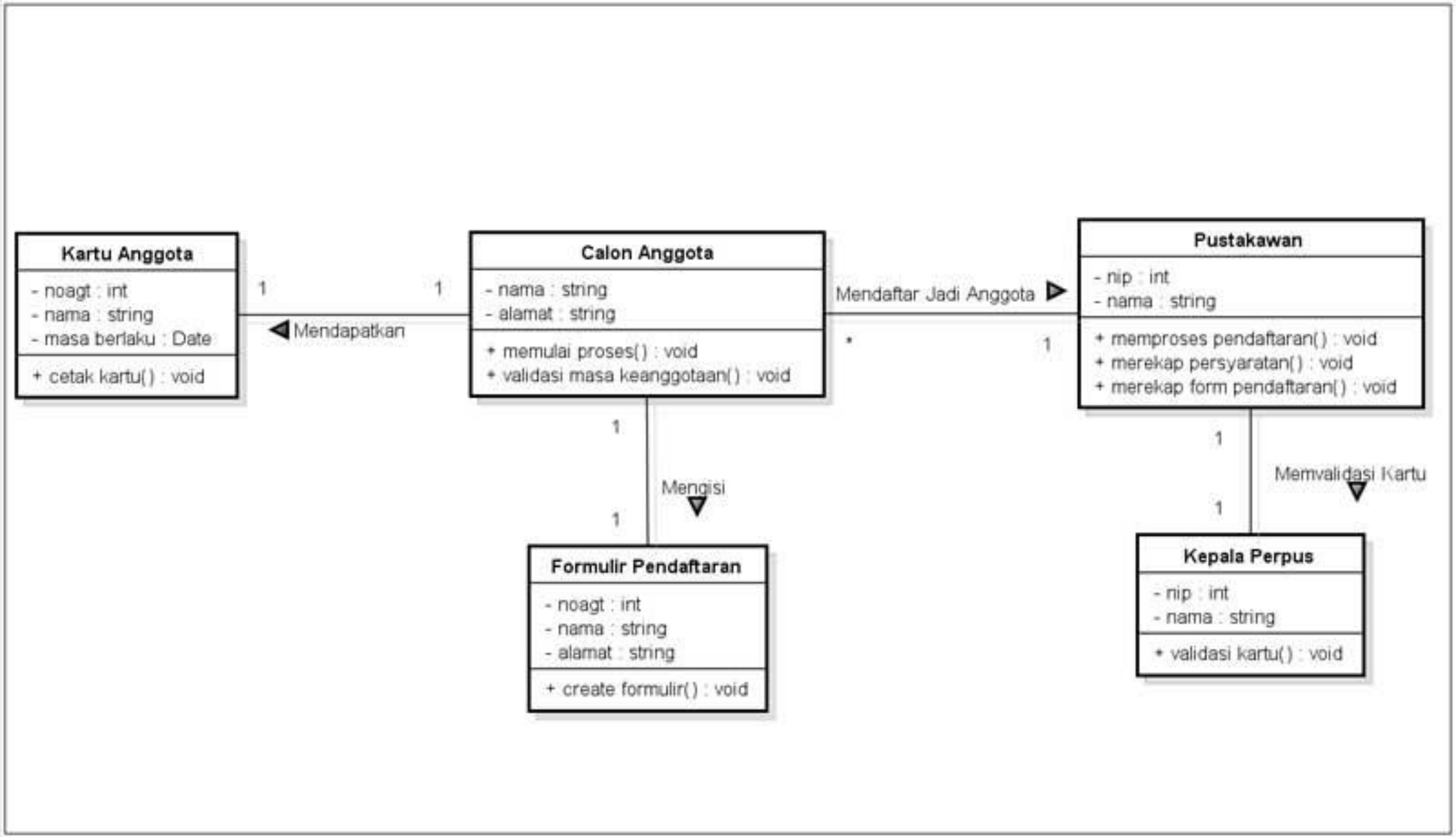
+
Public

-
Private

*
Protected

Nama Kelas
Atribut: tipe balikan
Methods: tipe balikan

Buat Sample Class di UML


Kartu Anggota
- noagt : int
- nama : string
- masa berlaku : Date
+ cetak kartu( ) : void
1
1
Mendapatkan
Calon Anggota
- nama : string
- alamat : string
+ memulai proses( ) : void
+ validasi masa keanggotaan( ) : void
Mendaftar Jadi Anggota
*
1
Pustakawan
- nip : int
- nama : string
+ memproses pendaftaran( ) : void
+ merekap persyaratan( ) : void
+ merekap form pendaftaran( ) : void
1
Mengisi
1
Formulir Pendaftaran
- noagt : int
- nama : string
- alamat : string
+ create formulir( ) : void
1
Memvalidasi Kartu
1
Kepala Perpus
- nip : int
- nama : string
+ validasi kartu( ) : void

Beberapa Prinsip Perancangan Kelas

Encapsulation

Coupling

Cohesion

Code Duplication

Coupling	Keterikatan antar kelas
	Kelas berkomunikasi melalui antar muka yang telah didefinisikan dengan baik
	Yang baik : *loose coupling*
	Perubahan pada satu kelas tidak memiliki pengaruh yang besar pada kelas lain

Encapsulation	Digunakan untuk mengurangi efek coupling.
	Menegaskan pemisahan antara *what* dan *how* dengan membuat atribut sebagai private dan menggunakan fungsi acessor untuk mengakses atribut tersebut.

Tidak semua atribut dan perilaku perlu dikeluarkan /dipublikan

Kita bisa menyembunyikan details dari suatu class/object dari class/object lain

Untuk mengakses data yang bersifat private tersebut kita bisa menggunakan atribut atau methods yang bersifat public

- ✓ Enkapsulasi berarti mempaketkan atribut dan methods dalam suatu kelas dan detail yang tidak menyangkut pihak luar di sembunyikan

- ✓ Membungkus atau mengkapsulkan

Access Control Modifiers

Wilayah Akses	Public	Private	Protected	Default
Di kelas yang sama	√	√	√	√
Beda kelas, di package yang sama	√	X	√	√
Beda kelas, beda package, di kelas turunan	√	X	√	X
Beda kelas, beda package, tidak di kelas turunan	√	X	X	X

Bagaimana mengakses atribut Private?

Gunakan method yang memungkinkan Anda mengakses atribut private.

Method yang memungkinkan Anda untuk mendapatkan data adalah **Accessor**.

Method yang memungkinkan Anda untuk mengubah data adalah **Mutator**.

Fungsi Mutator

- Fungsi untuk mengubah property dari suatu objek.
- Mengubah nilai atau value dari sebuah atribut.

Fungsi Accessor

- Fungsi untuk mendapatkan property dari suatu objek.
- Mengembalikan nilai atau value dari suatu atribut.

Accessor

```
Public class Lingkaran
{
    private double radius;

    public Lingkaran(double r)
    {radius=r;}
    public double getRadius()
    {return radius;}
}
```

Mutator

```
Public class Lingkaran
{
    private double radius;

    public Lingkaran(double r)
    {radius=r;}
    public void setRadius(double r)
    {this.radius=r;}
}
```

Cohesion	Satu unit kode seharusnya selalu bertanggung jawab pada satu dan hanya satu tugas (task)
	Cohesion dapat diterapkan pada kelas dan method
	Satu method hanya bertanggung jawab pada satu dan hanya satu *well-defined task*.
	Setiap kelas harus merepresentasikan entitas tunggal, *well-defined* dalam domain problem.

```java
/**
* Main play routine. Loops until end of play.
*/
public void play()
{
	System.out.println();
	System.out.println("Welcome to The World of Zuul!");
	System.out.println("Zuul is a new, incredibly boring adventure game.");
	System.out.println("Type 'help' if you need help.");
	System.out.println();
	System.out.println(currentRoom.getLongDescription());

	// Enter the main command loop. Here we repeatedly read
	// commands and execute them until the game is over.

	boolean finished = false;
	while (! finished) {
		Command command = parser.getCommand();
		finished = processCommand(command);
	}
	System.out.println("Thank you for playing. Good bye.");
}
```

```java
/**
* Main play routine. Loops until end of play.
*/
public void play()
{
        System.out.println();
        System.out.println("Welcome to The World of Zuul!");
        System.out.println("Zuul is a new, incredibly boring adventure game.");
        System.out.println("Type 'help' if you need help.");
        System.out.println();
        System.out.println(currentRoom.getLongDescription());

        // Enter the main command loop. Here we repeatedly read
        // commands and execute them until the game is over.

        boolean finished = false;
        while (! finished) {
                Command command = parser.getCommand();
                finished = processCommand(command);
        }
        System.out.println("Thank you for playing. Good bye.");
}
```

Task utama fungsi play()

```java
/**
* Main play routine. Loops until end of play.
*/
public void play()
{
        System.out.println();
        System.out.println("Welcome to The World of Zuul!");
        System.out.println("Zuul is a new, incredibly boring adventure game.");
        System.out.println("Type 'help' if you need help.");
        System.out.println();
        System.out.println(currentRoom.getLongDescription());

        // Enter the main command loop. Here we repeatedly read
        // commands and execute them until the game is over.

        boolean finished = false;
        while (! finished) {
                Command command = parser.getCommand();
                finished = processCommand(command);
        }
        System.out.println("Thank you for playing. Good bye.");
}
```

Task tambahan fungsi play()

Task utama fungsi play()

```java
/**
* Main play routine. Loops until end of play.
*/
public void play()
{
        printWelcome();
        // Enter the main command loop. Here we repeatedly read
        // commands and execute them until the game is over.

        boolean finished = false;
        while (! finished) {
                Command command = parser.getCommand();
                finished = processCommand(command);
        }
        System.out.println("Thank you for playing. Good bye.");
}
```

```java
/**
* Print out the opening message for the player.
*/
private void printWelcome()
{
        System.out.println();
        System.out.println("Welcome to The World of Zuul!");
        System.out.println("Zuul is a new, incredibly boring adventure game.");
        System.out.println("Type 'help' if you need help.");
        System.out.println();
        System.out.println(currentRoom.getLongDescription());
}
```

Code Duplication	Pertanda rancangan kelas yang tidak baik
	Problem : perubahan di satu bagian harus diikuti dengan bagian lain untuk menghindarkan inkonsistensi.
	Akibat : pemeliharaan menjadi mahal

```java
public class Game
{
	// ... some code omitted...
	/**
	* Print out the opening message for the player.
	*/
	private void printWelcome()
	{
		System.out.println();
		System.out.println("Welcome to the World of Zuul!");
		System.out.println("Zuul is a new, incredibly boring adventure game.");
		System.out.println("Type 'help' if you need help.");
		System.out.println();
		System.out.println("You are " + currentRoom.getDescription());
		System.out.print("Exits: ");
		if(currentRoom.northExit != null)
			System.out.print("north ");
		if(currentRoom.eastExit != null)
			System.out.print("east ");
		if(currentRoom.southExit != null)
			System.out.print("south ");
		if(currentRoom.westExit != null)
			System.out.print("west ");
		System.out.println();
	}

	// ... some code omitted...
```

```java
private void goRoom(Command command)
{
	if(!command.hasSecondWord()) {
		System.out.println("Go where?");
		 return;
	}
	String direction = command.getSecondWord();
	Room nextRoom = null;
	if(direction.equals("north"))
		nextRoom = currentRoom.northExit;
	if(direction.equals("east"))
		nextRoom = currentRoom.eastExit;
	if(direction.equals("south"))
		nextRoom = currentRoom.southExit;
	if(direction.equals("west"))
		nextRoom = currentRoom.westExit;
	if (nextRoom == null)
		System.out.println("There is no door!");
	else {
		currentRoom = nextRoom;
		System.out.println("You are " + currentRoom.getDescription());
		System.out.print("Exits: ");
		if(currentRoom.northExit != null)
			System.out.print("north ");
		if(currentRoom.eastExit != null)
			System.out.print("east ");
		if(currentRoom.southExit != null)
			System.out.print("south ");
		if(currentRoom.westExit != null)
			System.out.print("west ");
		System.out.println();
		}
	}
}
```

```java
public class Game
{
	// ... some code omitted...
	/**
	* Print out the opening message for the player.
	*/
	private void printWelcome()
	{
		System.out.println();
		System.out.println("Welcome to the World of Zuul!");
		System.out.println("Zuul is a new, incredibly boring adventure game.");
		System.out.println("Type 'help' if you need help.");
		System.out.println();
		System.out.println("You are " + currentRoom.getDescription());
		System.out.print("Exits: ");
		if(currentRoom.northExit != null)
			System.out.print("north ");
		if(currentRoom.eastExit != null)
			System.out.print("east ");
		if(currentRoom.southExit != null)
			System.out.print("south ");
		if(currentRoom.westExit != null)
			System.out.print("west ");
		System.out.println();
	}

	// ... some code omitted...
```

Duplikasi

```java
private void goRoom(Command command)
{
	if(!command.hasSecondWord()) {
		System.out.println("Go where?");
		 return;
	}
	String direction = command.getSecondWord();
	Room nextRoom = null;
	if(direction.equals("north"))
		nextRoom = currentRoom.northExit;
	if(direction.equals("east"))
		nextRoom = currentRoom.eastExit;
	if(direction.equals("south"))
		nextRoom = currentRoom.southExit;
	if(direction.equals("west"))
		nextRoom = currentRoom.westExit;
	if (nextRoom == null)
		System.out.println("There is no door!");
	else {
		currentRoom = nextRoom;
		System.out.println("You are " + currentRoom.getDescription());
		System.out.print("Exits: ");
		if(currentRoom.northExit != null)
			System.out.print("north ");
		if(currentRoom.eastExit != null)
			System.out.print("east ");
		if(currentRoom.southExit != null)
			System.out.print("south ");
		if(currentRoom.westExit != null)
			System.out.print("west ");
		System.out.println();
		}
	}
}
```

```java
public class Game
{
	// ... some code omitted...
	/**
	* Print out the opening message for the player.
	*/
	private void printWelcome()
	{
		System.out.println();
		System.out.println("Welcome to the World of Zuul!");
		System.out.println("Zuul is a new, incredibly boring adventure game.");
		System.out.println("Type 'help' if you need help.");
		System.out.println();



	}

	// ... some code omitted...
```

```java
private void goRoom(Command command)
{
	if(!command.hasSecondWord()) {
		System.out.println("Go where?");
		 return;
	}
	String direction = command.getSecondWord();
	Room nextRoom = null;
	if(direction.equals("north"))
		nextRoom = currentRoom.northExit;
	if(direction.equals("east"))
		nextRoom = currentRoom.eastExit;
	if(direction.equals("south"))
		nextRoom = currentRoom.southExit;
	if(direction.equals("west"))
		nextRoom = currentRoom.westExit;
	if (nextRoom == null)
		System.out.println("There is no door!");
	else {
		currentRoom = nextRoom;



		}
	}
}
```

```
public class Game
{
    // ... some code omitted...
    /**
    * Print out the opening me
    */
    private void printWelcome
    {
        System.out.printl
        System.out.printl
        System.out.printl
        System.out.printl
        System.out.printl
```

```
private void printLocationInfo()
    {
        System.out.println("You are " + currentRoom.getDescription());
        System.out.print("Exits: ");
        if(currentRoom.northExit != null)
            System.out.print("north ");
        if(currentRoom.eastExit != null)
            System.out.print("east ");
        if(currentRoom.southExit != null)
            System.out.print("south ");
        if(currentRoom.westExit != null)
            System.out.print("west ");
        System.out.println();
    }
```

```
    }

    // ... some code omitted...
```

```
private void goRoom(Command command)
{
    if(!command.hasSecondWord()) {
        System.out.println("Go where?");
        return;
```

```
        }
    }
}
```

```java
public class Game
{
	// ... some code omitted...
	/**
	* Print out the opening message for the player.
	*/
	private void printWelcome()
	{
		System.out.println();
		System.out.println("Welcome to the World of Zuul!");
		System.out.println("Zuul is a new, incredibly boring adventure game.");
		System.out.println("Type 'help' if you need help.");
		System.out.println();
		printLocationInfo();

	}

	// ... some code omitted...
```

```java
private void goRoom(Command command)
{
	if(!command.hasSecondWord()) {
		System.out.println("Go where?");
		 return;
	}
	String direction = command.getSecondWord();
	Room nextRoom = null;
	if(direction.equals("north"))
		nextRoom = currentRoom.northExit;
	if(direction.equals("east"))
		nextRoom = currentRoom.eastExit;
	if(direction.equals("south"))
		nextRoom = currentRoom.southExit;
	if(direction.equals("west"))
		nextRoom = currentRoom.westExit;
	if (nextRoom == null)
		System.out.println("There is no door!");
	else {
		currentRoom = nextRoom;
		printLocationInfo();

		}
	}
}
```

Masukkan Bilangan : 9

Menghitung Luas Lingkaran
Bilangan yang diinputkan = 9
Luas Lingkaran = xxxxxx

Menghitung Luas Segitiga
Bilangan yang diinputkan = 9
Luas Segitiga = xxxxxx