LAPORAN TUGAS BESAR I

IF2032 Pemrograman Berorientasi Objek

Simulasi Petani di Desa Opeh

<Opeh>

Dipersiapkan oleh:

Kelompok 07 (”Opeh”)

Ryan Setiadi (13506094)

Muhammad Adinata (13509022)

I Nyoman Prama Pradnyana (13509032)

Novan Parmonangan Simanjuntak (13509034)

Gurun Nevada Dharan (13509076)

Septu Jamasoka (13509080)

Nugraha (13509096)

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Sekolah Teknik Elektro dan Informatika ITB** | Nomor Dokumen | Halaman |
| *IF2032-TB1-<07>-<K2>* | *<jml hlm>* |
| *28 Februari 2011* |

Daftar Isi

Update daftar isi di bawah ini.

[1 Ringkasan 3](#_Toc287857359)

[2 Penjelasan Tambahan Spesifikasi Tugas 3](#_Toc287857360)

[2.1 <Spesifikasi Fitur Tambahan 1> 3](#_Toc287857361)

[2.2 <Spesifikasi Fitur Tambahan 2> 3](#_Toc287857362)

[3 Diagram Kelas 3](#_Toc287857363)

[4 Penjelasan Kelas 3](#_Toc287857364)

[5 Program Utama 3](#_Toc287857365)

[6 Algoritma-Algoritma Menarik 3](#_Toc287857366)

[6.1 <Algoritma 1> 3](#_Toc287857367)

[6.2 <Algoritma 2> 4](#_Toc287857368)

[7 Data Test 4](#_Toc287857369)

[7.1 <Data Test 1> 4](#_Toc287857370)

[7.2 <Data Test 2> 4](#_Toc287857371)

[8 Gameplay 4](#_Toc287857372)

[9 Pembagian Kerja dalam Kelompok 4](#_Toc287857373)

[10 Lampiran 4](#_Toc287857374)

[10.1 Notulen Rapat 4](#_Toc287857375)

[10.2 Log Activity Anggota Kelompok 4](#_Toc287857376)

[10.3 <Lain-Lain> 4](#_Toc287857377)

# Ringkasan

Tugas Besar I ini adalah pengimplementasian penggunaan pemrogramman berorientasi objek (OOP) dengan bahasa C++. Adapun yang menjadi tema tugas kali ini adalah Simulasi Petani di Desa Opeh (spesifikasi terlampir). Untuk tugas kali ini tidak menggunakan IDE (Integrated Development Environment) apapun.

Pada laporan kali ini akan membahas tentang keberjalan Tugas Besar I baik itu spesifikasi, penggunaan kelas, beberapa algoritma menarik, pengujiannya dan kesimpulan dari pembuatan tugas kali ini.

Setelah berhasil menyelesaikan tugas besar kali ini, didapat kesimpulan berupa pengerjaan tugas yang dilakukan secara bertahap dimulai dari (dalam C++) pembuatan rancangan kelas, kemudian dilanjurkan dengan implementasi kelas hingga selesai. Hal ini menjadikan pembuatan tugas besar menjadi terurut dengan baik sehingga tugas yang dihasilkan pun jelas dan sesuai dengan spesifikasi.

# Penjelasan Tambahan Spesifikasi Tugas

Tugas Besar I ini memiliki beberapa spesifikasi. Berikut akan dijelaskan mengenai beberapa spesifikasi yang kiranya perlu untuk dijelaskan kembali.

## Musim ”Season”

Fitur ini perupakan Fitur Bonus yang diimplementasikan ke dalam Tugas Besar I ini. Fitur ini menyediakan kondisi dimana terdapat musim dan sebagai aplikasinya tanaman dibuat berkategori sehingga tanaman hanya dapat tumbuh pada musim-musim tertentu saja. Adapun yang menjadi musim pada Tugas ini adalah musi, SPRING, SUMMER, dan FALL.

Fitur musim ini dijadikan atribut pada kelas Time. Nama yang digunakan saat pengimplementasian adalah season dan pengecekan musim adalah cseason. Pengambilan season ini digunakan dalam tipe integer sedangkan untuk cseason digunakan tipe boolean. Ketika dalam rentang 30 hari, nilai cseason diberikan false dengan tujuan agar selama 30 hari tersebut, season tidak berubah. Kemudian setelah rentang 30 hari maka cseason akan bernilai true dan cseason akan melanjutkan perintah berlanjut ke next season. Fungsi nsxt season ini kemudian dimplementasikan dengan nilai dan dicari kelipatan tiga (season dalam satu tahun ada 3 per 30 hari) kemudian ditambah satu. Jadi nilai pada season akan terus bertambah dan penentuan season selanjutnya dengan membagi nilai season dan menambahkannya dengan 1 sehingga didapat season baru yang berada pada tanggal 1 di bulan yang baru. Untuk pertama kali memainkan game ini, nilai season diberikan nilai 1. Sedangkan untuk perintah pengembalian season diberikan fungsi get season.

Pengaruh dari season ini kemudian digunakan dalam kelas grid\_plant. Pemanggila fungsi dilakukan sesuai dengan kelas Time dan dikondisikan tanaman mana yang dapat tumbuh pada musim SPRING, SUMMER, FALL.

## Tanaman Re-Plant

Fitur Tanaman Re-Plant ini juga merupakan bonus yang telah diimplementasikan ke dalam Tugas Besar I. Fitur ini mengkondisikan dimana terdapat tanaman yang setelah dipanen tanaman tidak langsung hilang. Hanya titik panennya saja yang berkurang setiap kali panen.

Fitur Tanaman Re-Plant ini dijadikan fungsi langsung dalam kelas grid\_Plant. Dalam fungsi ini dicek apakah tanaman dapat dipanen ulang atau tidak. Dalam kelas grid\_Plant Fitur re Plant ini dijadikan fungsi yang bernama panenBerulang. PanenBerulang ini bertipe boolean dimana jika suatu tanaman pada fungsi panenBerulangnya di berikan nilai false, maka tanaman tersebut tidak dapat tumbuh kembali, namun jika diberikan nilai true, maka tanaman tersebut dapat tumbuh kembali. Tanaman yang telah dipanen akan kembali ke titik dewasa dan emnunggu beberapa waktu agar dapat dianen lagi.

## Kejadian Alam

Untuk fitur ini juag merupakan bonus yang diimplementasikan ke dalam Tugas Besar I. Fitur ini menambahkan beberapa kejadian alam yang terjadi secara acak dan memilii dampak yang berbeda-beda setiap kejadiannya. Terdapat beberapa kejadian alam yang digunakan dalam tugas kali ini antara lain Normal, Raining, dan Tornado.

Fitur Kejadian Alam ini dijadikan atribut doWeather dan setWeather pada kelas World dengan nama weather. Pertama, weather ini disetting dengan nilai 0. Kemudian dengan menggunakan atribut setWeather akan dirandom. Setelah perandoman (antara 0 sampai 100) akan dikemblikan lagi ke doWeather dan dicek. Jika nilainya kurang dari 50, maka akan normal, kemudian jika diantara 50 dan 80 maka weather akan berstatus Raining kemudian antara 80 sampai 100 akan berstatus tornado.

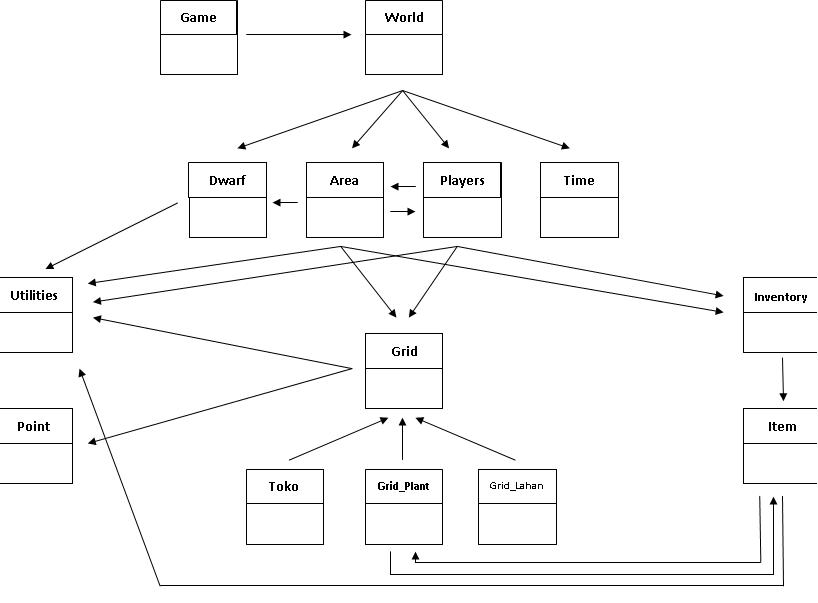
## Memakan Hasil Panen

Fitur Memakan Hasil Panen ini juga merupakan bonus yang telah diimplementasikan ke dalam Tugas Besar I. Fitur ini memungkinkan pemain dapat memakan hasil panen dan setiap kali memakan akan menimbulkan efek yang berbeda-beda tergantung dari hasil panen yang dimakannya. Item yang bisa dimakan adalah buah (hasil panen).

Fitur ini dijadikan atribut pada kelas Player dengan nama atribut eat. Atribut ini akan memanggil atribut cekslot pada kelas Inventory kemudian mengurangi jumlah inventory pada slot. Setelah pengurangan dilanjutkan dengan melempar fungsi getbuah dan get status. Untuk lamanya efek berjalan dilempar fungsi getefektime yang mengukur lamanya waktu efek.

Sedangkan untuk penjelasan dan keterangan efek dari item yang dimakan diimplementasikan pada kelas Item. Pada kelas item akan disetting mengenai efek yang terjadi pada player ketika memakannya. Adapun efek yang ditawarkan adalah mengenai daya kerja perjamnya. Terdapat buah yang dapat menjadikan player menambah kesempatan dalam menambha daya kerjanya dalam satu jam, disamping itu terdapat buah yang justru mengurangi daya kerja player per jamnya.

# Diagram Kelas



Gambar 3.1 Alur Diagram Kelas

# Penjelasan Kelas

|  |  |  |
| --- | --- | --- |
| **Nama Kelas** | **Member Data** | **Method** |
| **World** | **private :**  Time\* time;  Player\* player;  Area\* area[3];  Kurcaci\* kurcaci[3];  int weather; | **private :**  **void doWeather();**  **public :**  World();  World(World&);  virtual ~World();  World& operator=(const World&);  friend ostream& operator<<(ostream&, const World&);  static void save(const string&, const World&);  static World\* load(const string&);  void setWeather();  Area\* getArea(int);  dwarf\* getdwarf(int);  time\* gettime();  Player\* getPlayer();  void dwarfWork(); |
| **Game** | **public :**  string Perintah; | **public :**  Game();  Game(string);  Game(const Game&);  virtual ~Game();  void run();  **private :**  void getPerintah() const;  void doPerintah(string); |
| **Area** | **private :**  Grid\* grid[10][10];  **public :**  const int typeArea; | **public :**  Area(int& tipe);  Area();  virtual ~Area();  Grid\* getGrid(Point); |
| **dwarf** | **private :**  point cpos;  int direction;  int status;  const int type; | Friend void nextmove(dwarf,dwarf,dwarf);  **public** :  dwarf(int);  ~dwarf();  bool dojob();  void wakeup();  void sleep();  int getstatus();  void performstatus();  void setposition(point);  point getposition();  void setdirection(int);  int getdirection(); |
| **Player** | **private :**  Grid \* curGrid;  Area \* curArea;  Inventory inventory;  int money;  int arahHadap;  int status;  string nama; | **public :**  Player();  Player(Inventory,int,string);  Player(const Player&);  virtual ~Player();  void setMoney();  int getMoney();  Grid \* getFrontGrid();  void setStatus(Item);  int getStatus();  void setName(string);  string getName();  void plow();  void slash();  void water();  void eat(int);  void put(int,int);  void move(int);  void harvest();  void sellItem(Item);  void buyItem(Item);  void teleport(area); |

1. **Penjelasan Kelas World**

Kelas **World** adalah kelas yang menggambarkan keseluruhan permainan pada suatu waktu.

*Member data* dan *method* pada kelas **World** yaitu :

1. ***Member data* (*private*)**
   1. Time\* time;
   2. Player\* player;
   3. Area\* area[3];
   4. Kurcaci\* kurcaci[3];
   5. int weather;
2. ***Member data* (*public*)**

Pada Kelas **World** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**
   * 1. void doWeather();

Menyesuaikan keadaaan world sesuai dengan hari itu. Dilakukan saat setWeather dilakukan dan berdasarkan tipe cuaca yang ditentukan attribute weather.

1. ***Method* (*public*)**
2. World(); //ctor
3. World(World&); //cctor
4. virtual ~World(); //dtor
5. World& operator=(const World&); //assignment operator
6. friend ostream& operator<<(ostream&, const World&);

Menampilkan kondisi world ke layar untuk dilihat oleh user.

1. static void save(const string&, const World&);

Menyimpan kondisi world ke file di pathFile.

1. static World\* load(const string&);

Mengembalikan pointer ke objek world sesuai file di pathFile

1. void setWeather();

Merandom cuaca untuk hari itu.

1. Area\* getArea(int);

Mengembalikan pointer ke area[i].

1. dwarf\* getDwarf(int);

Mengembalikan pointer ke dwarf[i].

1. time\* gettime();

Mengembalikan pointer ke time.

1. Player\* getPlayer();

Mengembalikan pointer ke player.

1. void dwarfWork();

Membuat seluruh kurcaci bekerja.

1. **Penjelasan Kelas Game**

Kelas **Game** adalah kelas yang mengenkapsulasi seluruh kelas yang ada. Kelas ini akan menerima masukkan dari user dan mengarahkannya ke kelas mana masukkan itu ditujukan

*Member data* dan *method* pada kelas **Game** yaitu :

1. ***Member data* (*private*)**
   1. string[] Perintah;
2. ***Member data* (*public*)**

Pada Kelas **Game** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**
   1. void getPerintah() const;

Menerima masukan perintah dari user untuk kemudian di-parsing.

* 1. void doPerintah(string);

Melakukan aksi yang sesuai dengan method yang tepat.

1. ***Method* (*public*)**
2. Game(); //ctor
3. virtual ~Game(); //dtor
4. void run();

Method yang dipanggil untuk memulai game.

1. **Penjelasan Kelas Area**

Kelas **Area** adalah kelas yang menggambarkan keseluruhan permainan pada suatu waktu.

*Member data* dan *method* pada kelas **Area** yaitu :

1. ***Member data* (*private*)**
   1. Grid\* grid[10][10];
2. ***Member data* (*public*)**
   1. const int typeArea;
3. ***Method* (*private*)**

Pada Kelas **Area** tidak memiliki ***Method* (*public*)**

1. ***Method* (*public*)**
2. Area(int& tipe); // cctor
3. Area(); // ctor
4. virtual ~Area(); // dtor
5. Grid\* getGrid(Point);

Mengembalikan pointer yang menunjuk ke grid[p.x][p.y].

* 1. Grid\* getGrid(int,int);

Mengembalikan pointer yang menunjuk ke grid[x][y]**.**

1. **Penjelasan Kelas dwarf**

Kelas dwarf merupakan kelas kurcaci,terdapat tiga objek dwarf/kurcaci berdasarkan type(ada tiga type, masing-masing type punya keahlian khusus) dari kurcaci, 0 untuk water, 1 untuk harvest, 2 untuk slash.

*Member data* dan *method* pada kelas **Dwarf** yaitu :

1. ***Member data (private)***
   1. point cpos;

merupakan current position dari suatu dwarf

* 1. Grid \* curGrid;

current Grid dari dwarf

* 1. Area[1] \* Field;

pointer ke lahan, digunakan untuk mencari langkah cerdas, nantinya diconvert dulu ke map

* 1. int direction;

arah hadap kurcaci, 1 untuk left, 2 untuk up,3 untuk right, 4 untuk down

* 1. int status;

status dari kurcaci, 1 for wakeup, 0 for sleep

* 1. const int type;

merupakan type dari objek kurcaci,0 for water,1 for harvest, //2 for slash.

* 1. int success;

status keberhasilan suatu dwarf, 1 for success, 0 for fail, dwarf dikatakan berhasil jika dia berhasil melakukan pekerjaan sesuai dengan keahliannya

1. ***Method (public)***
2. void dojob();

dwarf melakukan aksi, sesuai dengan typenya,di sini jika berhasil dilakukan maka nilai success akan di set menjadi 1

* 1. void wakeup();

mengaktifkan dwarf yang tertidur,setting status=1

* 1. void sleep();

menonaktifkan dwarf, setting status=0

* 1. int getstatus();

mengembalikan status dwarf

* 1. void performstatus();

menuliskan kondisi status kurcaci ke layar, formatnya <nomor\_kurcaci> <sleep/wake\_up>

* 1. void setposition(point p);

setting posisi dwarf , posisi diganti dengan p

* 1. point getposition();

mengembalikan current position dwarf

* 1. void setdirection(int nd);

setting direction dari dwarf berdasarkan nilai nd

* 1. int getdirection();

mengembalikan direction dari dwarf

* 1. void setsuccess0();

set success=0/belum melakukan pekerjaan

* 1. int getsuccess();

mendapatkan nilai dari current success

1. ***Method friend/ fungsi non-anggota***
   1. nextmove(dwarf,dwarf,dwarf)

Fungsi ini menetukan langkah berikutnya dari dwarf, sbg slh satu fungsi non-anggota, memakai class map untuk penyederhanaan masalah, di fungsi ini map diisi sesuai dengan lahan, lalu nantinya nilai ini dikembalikan lagi ke lahan/grid, di sini kita dalam menetukan pencarian langkah cerdas menggunakan algoritma bfs conditional, sehingga diperlukan ketiga posisi dwarf dalam menentukan langkah cerdas, tidak bisa satu per satu, ini alasan utama, nextmove menjadi fungsi non-anggota dari kelas dwarf.

1. **Penjelasan Kelas Player**
   * + 1. ***Member Private***
          1. Grid \* curGrid;

grid player berada

* 1. Area \* curArea;

area player berada

* 1. Inventory inventory;

inventory dari player

* 1. int money;

jumlah uang

* 1. int arahHadap;

arah hadap player, 1 = atas, 2 = kanan, 3 = bawah, 4 = kiri

* 1. int status;

status player akibat makanan yang dimakan

* 1. string nama;

nama dari player

* + - 1. ***Method Public***
         1. void setMoney(int);

Mengisi uang untuk player

* 1. int getMoney();

Mengambil nilai money pada Player

* 1. Grid \* getFrontGrid();

Mengambil grid di depan pemain dihitung dari curGrid dan arahHadap

* 1. void setStatus(Item);

Mengisi jenis status pada item

* 1. int getStatus();

Memeroleh informasi status dari player

* 1. void setName(string);

Mengisi nama dari player

* 1. string getName();

Mengambil nama dari player

* 1. void plow();

Prosedur mencangkul dengan proses memeroleh frontGrid Player, kemudian mengecek tipenya apakah merupakan lahan plantable, jika iya, maka fase akan berubah menjadi sudah dicangkul, jika tipe lahan plant, maka jika masih pada fase bibit, bibit akan hilang

* 1. void slash();

Prosedur memotong dengan proses memeroleh frontGrid Player, kemudian mengecek tipenya apakah merupakan tanaman, jika iya, maka tipe grid akan berubah menjadi lahan kosong, dan fase akan berubah menjadi tidak ada.

* 1. void water();

Prosedur menyiram dengan proses memeroleh frontGrid Player, kemudian mengecek tipenya apakah merupakan tanaman, jika iya, maka fase akan berubah menjadi sudah disiram

* 1. void eat(int);

Prosedur memakan item yang ada pada inventory dengan slot tertentu, kemudian mengecek efek yang terjadi, dan memberikan status pada player, serta melakukan perubahan aksi sesuai status yang dialami player

* 1. void put(int,int);

Prosedur untuk meletakkan barang dengan proses memeroleh frontGrid Player, kemudian mengecek tipenya apakah lahan atau selling box atau yang lain. Jika lahan, jumlah hanya bisa bernilai 1 dan tipe akan berubah menjadi tanaman, dan fase juga akan berubah menjadi bibit. Jika selling box, maka akan dipanggil prosedur sell yang artinya terjadi penjualan barang. Jika di grid selain tipe di atas, maka item akan hilang sesuai jumlah

* 1. void move(int);

Prosedur untuk pergerakan dari player dimana jika sedang menghadap atas dan ke kiri, maka prosedur akan membuat arah menjadi ke kiri dulu , baru bergerak ke kiri, dan jika sudah pada arah yang benar, maka player akan bergerak ke grid di depannya dengan pengecekan apakah grid bisa dilalui.

* 1. void harvest();

Prosedur untuk mengambil tanaman yang telah mencapai titik panen jika frontGrid merupakan tanaman dan telah mencapai titik panen. Tanaman kemudian akan menyesuaikan dengan tipe tanaman apakah langsung hilang atau bisa dipanen ulang. Jika bisa dipanen ulang, maka tanaman akan kembali ke fase tanaman dewasa. Jika tidak bisa, maka fase akan berubah menjadi 0, dan tipe grid akan menjadi lahan

* 1. void sellItem(Item);

Prosedur untuk menjual item, dimana uang akan bertambah sesuai dengan penjualan dan jenis barang yang dijual

* 1. void buyItem(Item);

Prosedur untuk membeli item, dimana uang akan berkurang sesuai dengan jumlah dan jenis barang yang dibeli

* 1. void teleport(Area);

Prosedur untuk berpindah area dari area yang satu ke area yang lain dengan ketentuan Rumah -> Lahan -> Toko dan sebaliknya Toko -> Lahan -> Rumah

1. **Penjelasan Kelas grid**

Kelas **Grid** merupakan kotak-kotak pembentuk area.

*Member data* dan *method* pada kelas **Grid** yaitu :

1. ***Member data* (*private*)**
   * + 1. Point posisi;
       2. int type;
       3. int fase;
2. ***Member data* (*public*)**

Pada Kelas **Grid** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**

Pada Kelas **Grid** tidak memiliki ***Method* (*public*)**

1. ***Method* (*public*)**
2. grid(); // ctor
3. grid(point,int,int); // cctor
4. grid(int,int,int); // cctor
5. grid(const grid&); // cctor
6. virtual ~grid(); // dtor
7. Point getPosisi() const; //

Mengembalikan point posisi dari grid.

1. int getType() const;

Mengembalikan tipe dari grid.

1. int getFase() const;

Mengembalikan fase dari grid.

1. boolean isPlantable() const;

Mengembalikan plantable dari grid.

1. void setType(int);

Mengeset tipe dari grid.

1. void setFase(int);

Mengeset fase dari grid.

1. void setPosisi(point);

Mengeset posisi dari grid.

* 1. ostream& operator<<();

Menulis grid ke layar

1. **Penjelasan Kelas grid\_lahan**

Kelas **Grid\_Lahan** merupakan kotak-kotak pembentuk area yang berupa lahan, turunan dari grid.

*Member data* dan *method* pada kelas **Grid** yaitu :

1. ***Member data* (*private*)**

Pada Kelas **Grid\_Lahan** tidak memiliki ***Member data* (*private*)**

1. ***Member data* (*public*)**

Pada Kelas **Grid\_Lahan** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**

Pada Kelas **Grid\_Lahan** tidak memiliki ***Method* (*public*)**

1. ***Method* (*public*)**
2. grid\_lahan(); //ctor
3. grid\_lahan(point,int,int); // cctor
4. grid\_lahan(const grid\_lahan&); //cctor
5. virtual ~grid\_lahan();//dtor
6. bool isWatered();

mengembalikan true jika sudah disiram.

1. void setCangkul();

pengubahan fase lahan setelah dicangkul

1. void setSiram();

pengubahan fase lahan setelah disiram

1. void setPut();

pengubahan fase lahan setelah ditanam bibit

1. **Penjelasan Kelas grid\_plant**

Kelas **Grid\_Plant** merupakan kotak-kotak pembentuk area yang berupa lahan, turunan dari grid.

*Member data* dan *method* pada kelas **Grid** yaitu :

1. ***Member data* (*private*)**
   * + 1. Point posisi;
       2. int type;
       3. int fase;
2. ***Member data* (*public*)**

Pada Kelas **Grid\_Plant** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**

Pada Kelas **Grid\_Plant** tidak memiliki ***Method* (*public*)**

1. ***Method* (*public*)**
   1. grid\_plant(); //ctor
   2. grid\_plant(point,int,int,int,int,int,int,int,int,bool); //cctor
   3. grid\_plant(const grid\_plant&); //cctor
   4. virtual ~grid\_plant(); //dtor
   5. is, set, get, sudah jelas dan tidak dijelaskan
   6. void grow();

Menyesuaikan fase tanaman setiap pergantian hari.

1. **Penjelasan Kelas inventory**

Inventory merupakan kelas yang menampung item-item

1. ***Member data* (*private*)**
   * + 1. int maxslot = 3;
       2. int slot[];
       3. int total[];
2. ***Member data* (*public*)**

Pada Kelas **Inventory** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**

Pada Kelas **Inventory** tidak memiliki ***Method* (*public*)**

1. ***Method* (*public*)**
2. ctor,dtor tidak dijelaskan
3. void addItem(item p, int jumlah);

Untuk menambah item ke inventory

1. void delete(item p, int jumlah);

Untuk mendelete item dari inventory

1. int cekslot(int nomorslot);
2. **Penjelasan Kelas Point**

Sudah jelas.

1. **Penjelasan Kelas Time**

Kelas Time yang digunakan di sini, berformatkan 60 minutes(0-59), tiap 10 minutes dilakukan act aksi(act adalah integer,private dari kelas Time), kemudian 24 jam(0-23),365 day(0-364),3 season(0-2), bisa diganti sesuai kebutuhan

* + - 1. ***Member data Private :***
         1. int minutes;

nilai dari current minutes

* + - * 1. int jam;

nilai dari current jam

* + - * 1. int day;

nilai dari current day

* + - * 1. int season;

nilai dari current season

* + - * 1. int act;

nilai dari current act

* + - 1. ***Method Public :***
         1. Time();

konstruktor,setting semua private member menjadi 0, kecuali act menjadi 6

* 1. virtual ~Time();

destruktor time

* 1. void next10minutes();

menambah 10 minutes ke depan, 10 minutes dihitung sebagai act aksi, di sini nilai default act=6

* 1. int getminutes();

mengembalikan nilai current minutes

* 1. void nextjam();

menambah 1 jam

* 1. int getjam();

mengembalikan nilai current jam

* 1. void nextseason();

set season ke season berikutnya

* 1. int getseason();

mendapatkan nilai current season

* 1. void nextday();
  2. int getday();
  3. void setzero();
  4. void setact(int=6);
  5. int getact();

1. **Penjelasan Kelas Toko**

Kelas Toko merupakan kelas turunan dari grid yang bersifat sebagai toko. Player dapat melakukan transaksi jual beli.

1. ***Member data* (*private*)**
   * + 1. int maxslot = 3;
       2. int slot[];
       3. int total[];
2. ***Member data* (*public*)**

Pada Kelas **Inventory** tidak memiliki ***Member data* (*public*)**

1. ***Method* (*private*)**

Pada Kelas **Inventory** tidak memiliki ***Method* (*public*)**

1. ***Method* (*public*)**
2. void listitem();

Melist semua item dan deskripsinya

# Program Utama

Isi dengan penjelasan mengenai algoritma program utama (driver).

# Algoritma-Algoritma Menarik

Pada pembuatan Tugas Besar ini, terdapat beberapa Algoritma yang dikiranya cukup menarik.

## <Algoritma 1>

Isi dengan algoritma yang dimaksud, di mana algoritma tersebut digunakan, dan mengapa menarik.

## <Algoritma 2>

Isi dengan algoritma yang dimaksud, di mana algoritma tersebut digunakan, dan mengapa menarik.

# Data Test

Isi dengan data test dan penjelasan yang diperlukan. Hal-hal yang harus dijelaskan untuk tiap data test:

* Fitur apa (saja) yang dites dengan data test ini
* Skenario untuk melakukan pengetesan
* Hasil apa yang seharusnya diberikan
* Penjelasan lain (jika ada)
* Data test-nya sendiri.

## <Data Test 1>

Isi dengan penjelasan mengenai data test.

## <Data Test 2>

Isi dengan penjelasan mengenai data test.

# Gameplay

Isi dengan penjelasan mengenai game yang buat. Sebagai contoh waktu yang diperlukan hingga suatu tanaman dewasa. Tambahkan sub bab seperlunya.

# Pembagian Kerja dalam Kelompok

Berikut adalah pembagian kerja kelompok :

|  |  |  |  |
| --- | --- | --- | --- |
| No. | NIM | Nama | Pembagian Kerja |
| 1. | 13509022 | Muhammad Adinata (PM) |  |
| 2. | 13509032 | I Nyoman Prama Pradnyana |  |
| 3. | 13509033 | Novan Parmonangan Simanjuntak |  |
| 4. | 13509076 | Gurun Nevada Dharan |  |
| 5. | 13509080 | Septu Jamasoka |  |
| 6. | 13509096 | Nugraha |  |

# Lampiran

## Notulen Rapat

Berikut adalah Notulensi Rapat yang telah dilakukan

|  |  |
| --- | --- |
| Tanggal : Kamis, 3 Maret 2011 | Catatan Rapat :  Menenukan keberjalanan program yang dimulai dari kelas Game dengan atribut run untuk memanggil kelas World dan menerima masukkan user.  Penentuan pembuatan Grid, bibit dan beberapa pembahasan mengenai kelas yang akan digunakan. |
| Tempat : Via Yahoo Massanger |
| Kehadiran Anggota :   1. Muhammad Adinata (13509022) 2. I Nyoman Prama Pradnyana (13509032) 3. Novan Parmonangan Simanjuntak (13509034) 4. Gurun Nevada Dharan (13509076) 5. Septu Jamasoka (13509080) 6. Nugraha (13509096) |

|  |  |
| --- | --- |
| Tanggal : Kamis, 6 Maret 2011 | Catatan Rapat :  Kami melakukan presentasi prosedur langkah-langkah dalam game. Kami juga menjelaskan deskripsi umum dari masing-masing kelas. Kami juga menjelaskan deskripsi dari masing-masing method dalam tiap kelas. Juga dijelaskan deskripsi dari masing-masing atribut dari kelas tersebut.  Asisten juga menanyakan tentang hubungan antara lahan dan tanaman. Asisten menyetakan bahwa kelas tanaman kurang cocok sebagai turunan dari kelas lahan.  Menanyakan kejelasan yang bisa dilakukan. |
| Tempat : Selasar Basis Data (Asistensi I) |
| Kehadiran Anggota :   1. Muhammad Adinata (13509022) 2. I Nyoman Prama Pradnyana (13509032) 3. Novan Parmonangan Simanjuntak (13509034) 4. Gurun Nevada Dharan (13509076) 5. Septu Jamasoka (13509080) 6. Nugraha (13509096) |

|  |  |
| --- | --- |
| Tanggal : Kamis, 11 Maret 2011 | Catatan Rapat :  Tipe Item belum lengkap  Diagram kelas dilengkapi. Ada legenda mengenai arah baik itu dari anak ke bapak.  Penjelasan menggunakan tabel  Luas lahan bebas  Getch() baru diimplementasikan |
| Tempat : Selasar Programming (Asistensi II) |
| Kehadiran Anggota :   1. Muhammad Adinata (13509022) 2. I Nyoman Prama Pradnyana (13509032) 3. Novan Parmonangan Simanjuntak (13509034) 4. Gurun Nevada Dharan (13509076) 5. Septu Jamasoka (13509080) 6. Nugraha (13509096) |

## Log Activity Anggota Kelompok

Berikut adalah LogActivity setiap anggota Kelompok :

|  |  |
| --- | --- |
| **Log Activity Tugas Besar I IF2032 – Pemrograman Berorientasi Objek** | |
| **Muhammad Adinata (13509022) (\*PM)** | |
| **Tanggal** | **Kegiatan** |
| 28 Februari 2011 |  |
| 1 Maret 2011 |  |
| 2 Maret 2011 |  |
| 3 Maret 2011 |  |
| 4 Maret 2011 |  |
| 5 Maret 2011 |  |
| 6 Maret 2011 |  |
| 7 Maret 2011 |  |
| 8 Maret 2011 |  |
| 9 Maret 2011 |  |
| 10 Maret 2011 |  |
| 11 Maret 2011 |  |
| 12 Maret 2011 |  |
| 13 Maret 2011 |  |
| 14 Maret 2011 |  |
| 15 Maret 2011 |  |
| 16 Maret 2011 |  |
| 17 Maret 2011 |  |
| 18 Maret 2011 |  |
| 19 Maret 2011 |  |
| 20 Maret 2011 |  |
| 21 Maret 2011 |  |
|  |

|  |  |
| --- | --- |
| **Log Activity Tugas Besar I IF2032 – Pemrograman Berorientasi Objek** | |
| **I Nyoman Prama Pradnyana (13509032)** | |
| **Tanggal** | **Kegiatan** |
| 28 Februari 2011 |  |
| 1 Maret 2011 |  |
| 2 Maret 2011 |  |
| 3 Maret 2011 |  |
| 4 Maret 2011 |  |
| 5 Maret 2011 |  |
| 6 Maret 2011 |  |
| 7 Maret 2011 |  |
| 8 Maret 2011 |  |
| 9 Maret 2011 |  |
| 10 Maret 2011 |  |
| 11 Maret 2011 |  |
| 12 Maret 2011 |  |
| 13 Maret 2011 |  |
| 14 Maret 2011 |  |
| 15 Maret 2011 |  |
| 16 Maret 2011 |  |
| 17 Maret 2011 |  |
| 18 Maret 2011 |  |
| 19 Maret 2011 |  |
| 20 Maret 2011 |  |
| 21 Maret 2011 |  |
|  |

|  |  |
| --- | --- |
| **Log Activity Tugas Besar I IF2032 – Pemrograman Berorientasi Objek** | |
| **Novan Parmonangan Simanjuntak (13509034)** | |
| **Tanggal** | **Kegiatan** |
| 28 Februari 2011 |  |
| 1 Maret 2011 |  |
| 2 Maret 2011 |  |
| 3 Maret 2011 |  |
| 4 Maret 2011 |  |
| 5 Maret 2011 |  |
| 6 Maret 2011 |  |
| 7 Maret 2011 |  |
| 8 Maret 2011 |  |
| 9 Maret 2011 |  |
| 10 Maret 2011 |  |
| 11 Maret 2011 |  |
| 12 Maret 2011 |  |
| 13 Maret 2011 |  |
| 14 Maret 2011 |  |
| 15 Maret 2011 |  |
| 16 Maret 2011 |  |
| 17 Maret 2011 |  |
| 18 Maret 2011 |  |
| 19 Maret 2011 |  |
| 20 Maret 2011 |  |
| 21 Maret 2011 |  |
|  |

|  |  |
| --- | --- |
| **Log Activity Tugas Besar I IF2032 – Pemrograman Berorientasi Objek** | |
| **Gurun Nevada Dharan (13509076)** | |
| **Tanggal** | **Kegiatan** |
| 28 Februari 2011 |  |
| 1 Maret 2011 |  |
| 2 Maret 2011 |  |
| 3 Maret 2011 |  |
| 4 Maret 2011 |  |
| 5 Maret 2011 |  |
| 6 Maret 2011 |  |
| 7 Maret 2011 |  |
| 8 Maret 2011 |  |
| 9 Maret 2011 |  |
| 10 Maret 2011 |  |
| 11 Maret 2011 |  |
| 12 Maret 2011 |  |
| 13 Maret 2011 |  |
| 14 Maret 2011 |  |
| 15 Maret 2011 |  |
| 16 Maret 2011 |  |
| 17 Maret 2011 |  |
| 18 Maret 2011 |  |
| 19 Maret 2011 |  |
| 20 Maret 2011 |  |
| 21 Maret 2011 |  |
|  |

|  |  |
| --- | --- |
| **Log Activity Tugas Besar I IF2032 – Pemrograman Berorientasi Objek** | |
| **Septu Jamasoka (13509080)** | |
| **Tanggal** | **Kegiatan** |
| 28 Februari 2011 |  |
| 1 Maret 2011 |  |
| 2 Maret 2011 |  |
| 3 Maret 2011 |  |
| 4 Maret 2011 |  |
| 5 Maret 2011 |  |
| 6 Maret 2011 |  |
| 7 Maret 2011 |  |
| 8 Maret 2011 |  |
| 9 Maret 2011 |  |
| 10 Maret 2011 |  |
| 11 Maret 2011 |  |
| 12 Maret 2011 |  |
| 13 Maret 2011 |  |
| 14 Maret 2011 |  |
| 15 Maret 2011 |  |
| 16 Maret 2011 |  |
| 17 Maret 2011 |  |
| 18 Maret 2011 |  |
| 19 Maret 2011 |  |
| 20 Maret 2011 |  |
| 21 Maret 2011 |  |
|  |

|  |  |
| --- | --- |
| **Log Activity Tugas Besar I IF2032 – Pemrograman Berorientasi Objek** | |
| **Nugraha (13509096)** | |
| **Tanggal** | **Kegiatan** |
| 28 Februari 2011 |  |
| 1 Maret 2011 |  |
| 2 Maret 2011 |  |
| 3 Maret 2011 |  |
| 4 Maret 2011 |  |
| 5 Maret 2011 |  |
| 6 Maret 2011 |  |
| 7 Maret 2011 |  |
| 8 Maret 2011 |  |
| 9 Maret 2011 |  |
| 10 Maret 2011 |  |
| 11 Maret 2011 |  |
| 12 Maret 2011 |  |
| 13 Maret 2011 |  |
| 14 Maret 2011 |  |
| 15 Maret 2011 |  |
| 16 Maret 2011 |  |
| 17 Maret 2011 |  |
| 18 Maret 2011 |  |
| 19 Maret 2011 |  |
| 20 Maret 2011 |  |
| 21 Maret 2011 |  |
|  |

## Lampiran

### Deskripsi Tugas Besar I

### Form Asistensi Tugas Besar I