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

[1 Ringkasan 3](#_Toc287857359)

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

[2.1 <Musim ”Season”> 3](#_Toc287857361)

[2.2 <Tanaman Re-Plant> 3](#_Toc287857362)

[2.3 <Kejadian Alam> 4](#_Toc287857362)

[2.4 <Memakan Hasil Panen> 4](#_Toc287857362)

[3 Diagram Kelas 5](#_Toc287857363)

[4 Penjelasan Kelas 6](#_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.

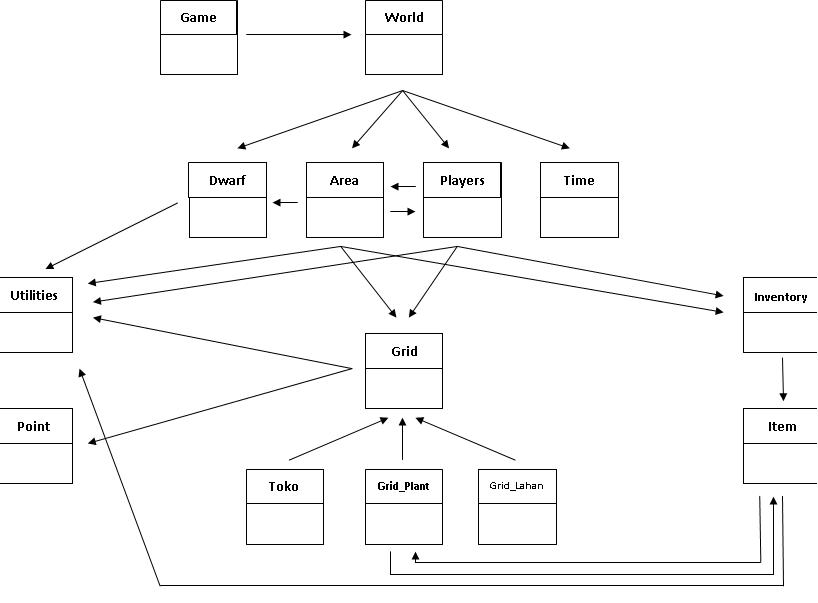
## Memakan Hasil Panen

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

Map

# Penjelasan Kelas

Berikut akan dijelaskan mengenai gambaran umum dari kelas-kelas yang digunakan dalam program ini.

## Penjelasan Kelas World

Kelas **World** adalah kelas yang menggabungkan seluruh kelas yang ada dan melakukan pemanggilan langsung terhadap kelas tersebut.

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

1. ***Member data* (*private*)**
   1. Time\* time

pointer ke objek Time

* 1. Player\* player

pointer ke objek Player yang telah diinstansiasi

* 1. Area\* area[3]

Array yang berisikan pointer yang menunjuk ke objek Area

* 1. Dwarf\* dwarf[3]

Array yang berisi pointer ke objek dwarf

* 1. int weather

Merupakan jenis cuaca yang ada pada hari itu. Diupdate setiap nextday()

* 1. do weather

melakukan efek cuaca pada hari itu. dilakukan saat setWeather dilakukan dan berdasarkan tipe cuaca yang ditentukan attribute weather.

1. ***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()

Konstruktor

1. World(string nama)

Parameter Player

1. World(World&)

Kopi Konstruktor

1. virtual ~World()

Destruktor

1. World& operator=(const World&)
2. 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.

## 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*)**

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

1. ***Member data* (*public*)**
2. string Perintah;
3. string paramStr[5];

Menyimpan parameter string

1. int paramInt[5];
2. ***Method* (*private*)**
3. void getNew(char\* kata, int done);
4. void getLoad(char\* kata, int done);
5. void getTeleport(char\* kata, int done);
6. void getArah (char\* kata, int done);
7. void getInventory (char\* kata, int done);
8. void getPlow(char\* kata, int done);
9. void getPut(char\* kata, int done);
10. void getSlash(char\* kata, int done);
11. void getWater(char\* kata, int done);
12. void getHarvest(char\* kata, int done);
13. void getWakeUp(char\* kata, int done);
14. void getSleep(char\* kata, int done);
15. void getStatus(char\* kata, int done);
16. void getSave(char\* kata, int done);
17. void getExit(char\* kata, int done);
18. void getStore(char\* kata, int done);
19. void getBuy(char\* kata,int done);
20. void getSell(char\* kata,int done);
21. void getCheat(char \* kata, int done);
22. void getEat(char \* kata, int done);
23. char\* parseLine();
24. ***Method* (*public*)**
25. World\* world;
26. Game();
27. void Run();
28. void getPerintah();
29. bool doPerintah();
30. virtual ~Game();

## Penjelasan Kelas Area

Kelas **Area** adalah kelas yang menyediakan akomodasi dari tampilan game.

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

1. ***Member data* (*private*)**
2. Grid \* grid[10][10];
3. Player\* player;
4. Dwarf\* dwarf[3];
5. ***Member data* (*public*)**
6. const int typeArea;
7. ***Method* (*private*)**

Pada Kelas **Area** tidak memiliki ***Merhod* (*private*)**

1. ***Method* (*public*)**
2. Player\* getPlayer();
3. Dwarf\* getDwarf(int);
4. void setPlayer(Player\*);
5. void setDwarf(int, Dwarf\*);
6. int getType();
7. Area(int tipe);
8. Area();
9. virtual ~Area();
10. Grid\* getGrid(Point);
11. Grid\* getGrid(int,int);
12. virtual void setGrid(Grid\*);
13. friend ostream& operator<<(ostream&, Area\*);

## Penjelasan Kelas dwarf

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*)**
2. const int type;
3. int status;
4. Point cpos;
5. int direction;
6. static map \*cmap;
7. deque<Point> \*dwarfdqp;
8. ***Member data* (*public*)**
9. Grid \*cGrid;
10. static Area \*Field;
11. ***Method* (*private*)**

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

1. ***Method* (*public*)**
2. Dwarf(int);
3. virtual ~Dwarf();
4. static void setDefault(Dwarf&,Dwarf&,Dwarf&);
5. int getType();
6. bool doJob();
7. void wakeUp();
8. void sleep();
9. int getStatus();
10. void performStatus();
11. void setPosition(int,int);
12. void setPosition(Point);
13. Point getPosition();
14. Point getFrontpoint();
15. void setDirection(int);
16. int getDirection();
17. static void setmap();
18. void setField();
19. void bfsdwarf();
20. void oneMove();
21. static void nextMove(Dwarf&,Dwarf&,Dwarf&);
22. Grid \* getFrontGrid();
23. friend ostream& operator<<(ostream& c, Dwarf\* p);
24. static map\* getMap();

## Penjelasan Kelas Player

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*)**
2. Grid \* curGrid;
3. Area \* curArea;
4. Inventory inventory;
5. int money;
6. int arahHadap;
7. int status;
8. char nama[32];
9. ***Member data* (*public*)**

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

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

Pada Kelas **Player** tidak memiliki ***Method* (*private*)**

1. ***Method* (*public*)**
2. Player();
3. Player(Area\*,int,string);
4. Player(const Player &);
5. virtual ~Player();
6. Player& operator= (const Player&);
7. void setMoney(int);
8. int getMoney();
9. Grid \* getFrontGrid();
10. void setStatus(Item);
11. void setStatus(int);
12. int getStatus();
13. void setName(string);
14. string getName();
15. Area \* getCurArea();
16. Grid \* getCurGrid();
17. Inventory \* getInventory();
18. void setFrontGrid(Grid\*);
19. void setCurGrid(Grid \* a);
20. void setArah(int);
21. void plow();
22. void slash();
23. void water();
24. void eat(int);
25. void put(int,int);
26. void move(int);
27. void harvest();
28. void sellItem(int,int);
29. void buyItem(string,int);
30. void teleport(Area\*);
31. friend ostream& operator<<(ostream&, Player\*);

## Penjelasan Kelas Time

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*)**
2. int minutes;
3. int jam;
4. int day;
5. int season;
6. bool cjam;
7. bool cday;
8. bool cseason;
9. int act;
10. int stime;
11. ***Member data* (*public*)**

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

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

Pada Kelas **Time** tidak memiliki ***Method* (*private*)**

1. ***Method* (*public*)**
2. Time();
3. virtual ~Time();
4. void next10Minutes(int);
5. void nextMinutes();
6. int getMinutes();
7. void nextJam();
8. int getJam();
9. void nextDay();
10. int getDay();
11. void nextSeason();
12. int getSeason();
13. void setZero();
14. bool iscJam();
15. bool iscDay();
16. bool iscSeason();
17. void setAct(int);
18. void performtime();
19. int getSTime();

## Penjelasan Kelas Utilities

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*)**
2. int color;
3. stack<int> oldFG;
4. stack<int> oldBG;
5. static Utilities\* utilities;
6. ***Member data* (*public*)**

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

1. ***Method* (*private*)**
2. Utilities();
3. void draw();
4. ***Method* (*public*)**
5. static Utilities& getInstances();
6. void setColor(int,int);
7. void setBG(int);
8. void setFG(int);
9. void resetColor();
10. void resetBG();
11. void resetFG();
12. void destroy();
13. void gotoxy(int,int);
14. int wherex();
15. int wherey();
16. void printPeta(ostream& c);
17. void printRumah (ostream& c);

## Penjelasan Kelas Point

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*)**
2. int x;
3. int y;
4. ***Member data* (*public*)**

Pada Kelas **Point** tidak memiliki ***Member data* (*private*)**

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

Pada Kelas **Point** tidak memiliki ***Method* (*private*)**

1. ***Method* (*public*)**
2. Point();
3. Point(int,int);
4. Point(const Point &);
5. ~Point();
6. Point& operator= (const Point &);
7. int getX();
8. void setX(int);
9. int getY();
10. void setY(int);
11. bool operator< (Point);
12. bool operator> (Point);
13. bool operator== (Point);
14. void setLeft();
15. void setUp();
16. void setRight();
17. void setDown();
18. Point getLeft();
19. Point getUp();
20. Point getRight();
21. Point getDown();
22. friend ostream& operator<<(ostream&,Point\*);
23. friend ostream& operator<<(ostream&,Point&);

## Penjelasan Kelas Inventory

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*)**
2. int maxslot;
3. int temp;
4. Item\* slot[3];
5. int total[3];
6. ***Member data* (*public*)**

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

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

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

1. ***Method* (*public*)**
2. Inventory();
3. Inventory(const Inventory&);
4. Inventory& operator= (const Inventory&);
5. virtual ~Inventory();
6. int listItem();
7. void addItem(const string& name, int jumlah);
8. void addItem(Grid\_Plant\* tanaman, int jumlah);
9. void deleteItem(int, int);
10. bool cekSlot(int);
11. Item\* getSlot(int);
12. int getJumlah(int noslot);
13. void listDescription();

## Penjelasan Kelas Item

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*)**
2. int efekBuah;
3. int costBuy;
4. int costSell;
5. int costSellBibit;
6. int typeTanaman;
7. int typeBuah;
8. int IDitem;
9. bool bibit;
10. char nama[32];
11. ***Member data* (*public*)**

Pada Kelas **Item** tidak memiliki ***Member data* (*private*)**

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

Pada Kelas **Item** tidak memiliki ***Method* (*private*)**

1. ***Method* (*public*)**
2. Item ();
3. virtual ~Item ();
4. static Item\* makeBibit(const string&);
5. static Item\* makeBuah(const Grid\_Plant\*);
6. bool operator==(const Item&);
7. int getCostBuy() const;
8. int getCostSell()const;
9. int getTipeBuah() const;
10. int getTipeTanaman() const;
11. int getEfekBuah() const;
12. int getIDitem() const;
13. string getNama() const;//untuk melakukan deskripsi tiap item
14. bool isBibit();
15. void setEfekBuah(int);
16. void setCostBuy(int);
17. void setCostSell(int);
18. void setCostSellBibit(int);
19. void setTypeTanaman(int);
20. void setTypeBuah(int);
21. void setIDitem(int);
22. void setBibit(bool);
23. void setNama(string);

## Penjelasan Kelas Grid

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*)**
2. Point posisi;
3. int type;
4. int fase;
5. ***Member data* (*public*)**

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

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

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

1. ***Method* (*public*)**
2. Grid();
3. Grid(Point,int,int);
4. Grid(int,int,int);
5. Grid(const Grid&);
6. virtual ~Grid();
7. Point getPosisi() const;
8. int getType() const;
9. int getFase() const;
10. bool isPlantable() const;
11. virtual void grow(int);
12. void setType(int);
13. void setFase(int);
14. void setPosisi(Point);

## Penjelasan Kelas Toko

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*)**
2. ***Member data* (*public*)**

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

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

Pada Kelas **Toko** tidak memiliki ***Method* (*private*)**

1. ***Method* (*public*)**
2. Toko();
3. Toko(const Toko&);
4. virtual ~Toko();
5. void listItem();

Melist semua item dan deskripsinya

## Penjelasan Kelas Grid Plant

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*)**
2. Int typeTanaman;
3. int cost;
4. int season;
5. int happyMeter;
6. int titikDewasa;
7. int titikPanen;
8. int umur;
9. bool panenBerulang;
10. ***Member data* (*public*)**
11. ***Method* (*private*)**
12. ***Method* (*public*)**
13. Grid\_Plant();
14. Grid\_Plant(int);
15. Grid\_Plant(Point,int,int,int);
16. Grid\_Plant(const Grid\_Plant&);
17. virtual ~Grid\_Plant();
18. int getTypeTanaman() const;
19. int getCost() const;
20. int getHappyMeter() const;
21. int getTitikDewasa() const;
22. int getTitikPanen() const;
23. int getUmur() const;
24. int getSeason() const;
25. bool isWatered() const;
26. bool isBibit() const;
27. bool isDewasa() const;
28. bool isPanen() const;
29. bool isPanenBerulang() const;
30. void setCangkul();
31. void setSlash();
32. void setSiram();
33. void setPanen();
34. void setTypeTanaman(int);
35. void setHappyMeter(int);
36. void setCost(int);
37. void setTitikDewasa(int);
38. void setTitikPanen(int);
39. void setUmur(int);
40. void setPanenBerulang(bool);
41. void grow(int);
42. friend ostream& operator<<(ostream&, Grid\_Plant\*);

## Penjelasan Kelas Grid\_Lahan

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*)**
2. ***Member data* (*public*)**
3. ***Method* (*private*)**
4. ***Method* (*public*)**
5. Grid\_Lahan();
6. Grid\_Lahan(Point,int,int);
7. Grid\_Lahan();
8. Grid\_Lahan(Point,int,int);
9. virtual ~Grid\_Lahan();
10. void grow(int);
11. friend ostream& operator<<(ostream&, Grid\_Lahan\*);
12. void setCangkul();
13. void setSiram();
14. void setPut();

## Penjelasan Kelas Map

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*)**
2. char \*\*arr;
3. int sizer;
4. int sizec;
5. int t0;
6. int t1;
7. int t2;
8. ***Member data* (*public*)**
9. ***Method* (*private*)**
10. ***Method* (*public*)**
11. map();
12. ~map();
13. map(const map &);
14. map& operator=(const map&);
15. int getsizer();
16. int getsizec();
17. void setmapij(int,int,char);
18. char getvalij(int,int);
19. void setvalpoint(Point,char);
20. char getvalpoint(Point);
21. void performmap();
22. void sett0(int);
23. void sett1(int);
24. void sett2(int);
25. int gett0();
26. int gett1();
27. int gett2();

# Program Utama

Program main adalah program utama dari program ini. Pada dasarnya program ini hanya memamnggil kelas Game saja untuk menjalankan program ini karena kelas Game sudah mencakup seluruh program. Adapun yang menjadi method dalam program ini antara lain

* + 1. Game\* game = new Game()
    2. game->Run()
    3. delete game
    4. return 0

# Algoritma-Algoritma Menarik

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

## <Dwarf>

Adapun yang menarik dari algoritma dwarf adalah sbb.

## <Utilities>

Adapun yang menarik dari algoritma Utilities adalah sbb.

# Data Test

Berikut merupakan beberapa tes yang dilakukan pada program

## <Teleport Pemain>

Data test ini bertujuan untuk mengecek apakah player dapat melakukan teleport menuju ke tempat yang telah diinstrusikan. Kondisi telah ditentukan misalnya jika berada di rumah maka hanya bisa dilakukan teleport ke lahan, dan untuk lahan bisa ke rumah dan ke Toko (spesifikasi terlampir). Langkah-langkahnya adalah dengan mengetik

”teleport <nama\_area> #”

Contoh : teleport lahan #

Hasilnya : Player dapat berpindah ke lahan

## <Teleport Help>

Data test ini bertujuan untuk mengecek apakah method help pada teleport dapat dipanggil. Langkah-langkahnya adalah dengan mengetik

”teleport –help #”

Hasilnya : Keluar bantuan dalam menjalankan perintah teleport.

## <Langkah Pemain>

Data tes ini untuk mengecek apakah player dapat berpindah sesuai dengan jumlah langkah yang telah ditentukan. Langkah-langkahnya adalah dengan mengetik

”<right/left/up/down> <jumlah\_langkah> #”

Contoh : right 7 #

Hasilnya : Pemain berhasil berpindah sepanjang banyaknya langkah masukan seperti contoh adalah pemain akan berpindah ke kanan sebanyak 7 langkah.

## <Inventory>

Data tes ini dilakukan untuk mengecek apakah pemain dapat melihat benda-benda di dalam ranselnya. Langkah-langkahnya adalah mengetik

”inventory #”

Hasilnya : akan keluar daftar barang yang terdapat di dalam ransel dengan perincian menampilkan Nama Barang dan Jumlah Barang. Jika slot kosong maka akan dimunculkan tanda -

## <Pemain Baru>

Data tes ini dilakukan untuk mengecek apakah method penerima pemain baru dapat dilakukan. Langkah-Lagkahnya adalah dengan mengetik

”new <nama\_pemain> #”

Hasilnya : pemain dapat masuk ke pogram dan memainkan program. Nama pemain akan disimpan seingga dapat digunakan untuk yang lainnya.

## <Load Pemain>

Data tes ini dilakukan untuk mengecek apakah proses loading pemain yang telah disimpan sebelumnya dapat dilakukan. Langkah-langkahnya adalah dengan mengetik

”load <nama\_file> #”

Hasilnya : pemain dapat meload data yang telah tersimpan sebelumnya.

## <Mencagkul>

Data tes ini dilakukan untuk mengecek apakah proses pencangkulan lahan telah bisa dilakukan. Langkah yang dilakukan adalah dengan mengetik

”plow #”

Hasilnya : lahan dapat dicangkul diamati dengan perubahan warna pada lahan

## <Menanam>

Data tes ini dilakukan untuk mengecek apakah player bisa menanam. Proses menanam ini hanya bisa dilakukan pada lahan yang telah dicangkul terlebih dahulu. Langkah-langkahnya adalah dengan mengetik

”put <nomor\_slot> 1 #”

Contoh : put 0 1 #

Hasil : lahan yang telah dicangkul akan terdapat gambar bibit yang telah tertanam, ini menandakan bahwa proses menanam dapat dilakukan. Jika pada contoh adalah melakukan penanaman bibit pada slot 0 pada ransel sebanyak 1 biji (1 lahan).

## <Sabit>

Data tes ini dilakukan untuk mengecek apakah player dapat melakukan penyabitan terhadap tanaman untuk membersihkan lahan. Adapun spesifikasi sudah terlampir antara lain tentang penyabitan yang dilakukan terhadp biji, terhadap tanah basah dll. Adapun langkah yang dilakukan adalah

“slash #”

Hasil : tanaman pada lahan menjadi hilang dan terlihat lahan kosong.

## <Water>

Data tes ini dilakukan untuk mengecek apakah method water dapat dilakukan oleh player. Langkah-langkah yang dilakukan adalah

“water #”

Hasil : lahan yang telah disiram telah berubah warna, ini menunjukkan bahwa water telah berhasil dilakukan oleh pemain.

## <Memanen>

Data tes ini dilakukan untuk mengecek apakah method memanen dapat dilakukan pada program ini. Panen ini dapat dilakukan hanya ketika tanaman telah mencapai kondisi berbuah (spesifikasi terlampir) Langkah-langkah yang dilakukan adalah

“harvest #”

Hasil : panen berhasil dilakukan terlihat dari wujud tanaman yang berubah. Untuk tanaman yang dapt tumbuh kembali tanaman akan berubah menjadi muda sedangkan tanaman yang tidak dapat tumbuh akan menjadi lahan kosong.

## <Menjual>

Data tes ini dilakukan untuk mengecek apakah pemain dapat melakukan penjualan terhadp hasil panen. Langkah-langkah yang dilakukan adalah

“put <nomor\_slot> <jumlah> #

Contoh : put 0 5 #

Hasil : penjualan berhasil dilakukan damati dari berkurangnya jumlah slot dan bertambahnya uang.

## <Membangunkan Kurcaci>

Data tes ini dilakukan untuk mengecek apakah program dapat dilakukan perintah bangun terhadap kurcaci. Langkah-langkahnya adalah dengan mengetik

“wake\_up <nomor\_kurcaci>”

Contoh : wake\_up 1 #

Hasil : kurcaci melakukan aktifitas sesuai dengan jobnya. Pada contoh maka kurcaci kedua (bernomor 1) akan melakukan harvest terhadap tanaman yang sudah siap untuk dipanen.

## <Menidurkan Kurcaci>

Data tes ini dilakukan untuk mengecek apakah program dapat dilakukan perintah untuk mengistirahatkan kurcaci. Adapun langlkah-langkah yang digunakan adalah mengetik

“sleep <nomor\_kurcaci> #”

Contoh : sleep 1 #

Hasil : kurcaci yang sedang melakukan aktifitas akan terhenti dan berubah status menjadi tertidur.

## <Mengecek status Kurcaci>

Data tes ini dilakukan untuk mengecek apakah program dapat dilakukan pengecekan terhadap ketiga kurcaci. Langkah-langkah yang dialakukan adalah :

“status kurcaci #”

Hasil : akan tampil status ketiga kurcaci. Data yang ditampilkan adalah <nomor\_kurcaci> <sleep/wake\_up>

## <Menyimpan Data>

Data tes ini dilakukan untuk melakuakn pengecekan terhadap proses penyimpanan data. Langka yang dilakukan adalah

“save #”

Hasil : permainan akan disimpan dan dapat di load ketika awal mulai permainan.

## <Sleep>

Data tes ini dilakukan dalam pengecekan terhadap proses sleep pada program. Langkah-langkahnya adalah

“sleep #”

Hasil : pemain akan dikondisikan tertidur dan permainan dihentikan dalam selang waktu 1 hari dipermainan.

## <Exit>

Data tes ini digunakan untuk melakuakan pengetesan terhadap fungsi exit untuk keluaran program. Langkah yang dilakukan adalah mengetikkan

“exit #”

Hasil : program keluar bertahap dari inti ke world kemudian ke game exit.

## <Membeli Bibit>

Data tes ini digunakan untuk melakukan pengetesan terhadap proses pembelian bibit pada program. Langkah-langkah yang dilakukan adalah dengan mengetikkan

“buy <nama\_item> <jumlah> #”

Contoh : buy potato 2 #

Hasil : barang yang dibeli sudah ada pada inventory. Berdasarkan contoh maka pada inventory terdapat kentang dengan jumlah yang ditemtukan. Jika memang sydah ada, maka hanya jumlahnya saja ynag bertambah.

## <Menjual Bibit>

Data tes ini dilakukan untuk melakukan pengecekan terhadap method menjual bibit. Penjualan ini dilakukan hanya ketika berada di took. Langkah-langkah yang dilakukan adalah dengan mengetikkan

“sell <nomer\_slot> <jumlah> #”

Contoh : sell 1 3 #

Hasil : barang pada inventory slot 2 akan berkurang dan uang pemain bertambah.

# Gameplay

Game kami ini memiliki tema yang hampir sama dengan game yang cukup terkenal saat ini. Game tersebut adalah “Harvest Moon”. Karena tema game yang kami buat hampir sama dengan game Harvest Moon, maka Game Play dari game kami ini hampir sama dengan dengan Game Play dari “Harvest Moon”. Game yang kami buat ini, memiliki tiga latar tempat. Yakni rumah, toko, dan kebun. Rumah merupakan suatu area dimana pemain dapat menyimpan permainan, tidur, dan keluar dari permaian. Rumah merupakan tempat bagi player untuk istirahat dengan tujuan untuk memulihkan energi. Pemain dapat tidur menggunakan perintah sleep #. Ketika pemain tidur, hari pada permainan tetap berjalan. Energi sangat penting bagi keberlangsungan permainan. Energi dari pemain berkurang sesuai dengan aktivitas yang dilakukannya. Selain sebagai tempat memulihkan energi, rumah juga merupakan default tempat ketika permainan dimulai. Setelah permainan dimulai, maka permainan dapat di save di area rumah dengan menggunakan perintah save #. Pemain juga dapat keluar dari permainan dengan menggunakan perintah exit #

Berikut adalah penjelasan tentang Game Play dari Game kami ini. Ketika permainan mulai dijalankan, maka pertama akan diminta untuk memilih apakah new atau continue. Setelah user menginput pilihan, maka selanjutnya ditampilkan interface ke layar. Jika user menggunakan perintah new <nama\_pemain> #, maka selanjutnya akan ditampilkan interface dari game yang terbaru. Interface yang terbaru ini berupa karakter di dalam rumah. Jika digunakan perintah load <nama\_file> #, maka game akan dimulai dari kejadian terakhir saat game di save. Data tentang game adalah data pada saat game terakhir kali di save.

Jika user menggunakan perintah new <nama\_pemain> #, selanjutnya berada di rumah. Karater dapat bergerak di dalam rumah. Setelah puas di rumah, karater dapat pindah ke latar tempat lain dengan menggunakan perintah teleport <nama\_area> #. Karena pada game kami ini, karakter hanya dapat berpindah dari rumah ke lahan, maka diinput data teleport lahan #. Di lahan, karakter dapat melakukan beberapa kegiatan.

Kegiatan-kegiatan yang dapat dilakukan oleh karakter di lahan adalah:

- Berjalan

- Menyiram

- Menyabit

- Mencangkul

- Membuang/memakai barang

- Memanen tanaman

Seperti sudah dijelaskan sebelumnya, bahwa ketika karakter melakukan kegiatan, maka energinya akan berkurang sesuai dengan kegiatannya.

Di lahan pemain dapat menanam tanaman, tetapi sebelum tanaman ditanam terlebih dahulu plantable area harus dicangkul. Perintah yang dimasukkan untuk mencangkul adalah plow #. Daerah yang tercangkul adalah daerah di depan pemain. Jika pemain mencangkul daerah yang non-plantable, maka tidak terjadi perubahan dan muncul pesan kesalahan. Setelah tanah di plow, maka selanjutnya bibit dapat ditanaman pada tanah yang dicangkul dengan menggunakan perintah put <nomer\_slot> 1 #. Titik yang ditanami adalah titik yang berada tepat di hadapan pemain. Jika titik yang berada di depan pemain belum dicangkul, maka tidak terjadi perubahan dan muncul pesan kesalahan. Selain itu jumlah bibit berkurang (bibit dianggap dibuang). Sejalan dengan waktu, maka tanaman akan semakin besar dan terus bertumbuh. Setelah tanaman mencapai umurnya(mati), maka tanaman dapat dibersihkan menggunakan perintah slash #. Titik yang tersabit adalah titik yang berada tepat di hadapan pemain. Jika titik yang berada di depan pemain adalah area kosong, maka tidak terjadi perubahan dan muncul pesan kesalahan. Selain untuk tanaman mati, perintah ini juga dapat dilakukan terhadap tanaman yang masih hidup untuk membersihkan lahan.

Setelah bibit di tanam, maka selanjutnya tanaman perlu dirawat setiap hari. Perawatan tanaman ini berupa penyiraman tanaman. Tanaman dapat disiram oleh pemain dengan menggunakan perintah water #. Titik yang tersiram adalah titik yang berada tepat di hadapan pemain. Jika titik yang berada di depan pemain adalah *non-plantable area* atau area yang sudah disiram, maka tidak terjadi perubahan dan muncul pesan kesalahan.

Setelah beberapa lama waktunya, maka tanaman akan menghasilkan buah. Buah ini dapat dipanen oleh pemain dengan menggunakan perintah harvest #. Titik yang terpanen adalah titik yang berada tepat di hadapan pemain. Jika titik yang berada di depan pemain adalah area kosong atau tanaman yang belum mencapai titik panen, maka tidak terjadi perubahan dan muncul pesan kesalahan. Hasil panen akan masuk ke dalam ransel pemain. Jika ransel penuh, maka proses pemanenan tidak dapat dilakukan dan muncul pesan kesalahan. Isi ransel dapat dicek dengan mengetik perintah inventory #.

Setelah pemain memanen tanaman, maka pemain dapat memakan hasil tanaman. Tanaman dapat dimakan menggunakan perintah eat <no\_slot>. Setiap hasil panen yang dimakan akan memberikan efek yang berbeda.

Pada lahan perkebunan terdapat pula tiga kurcaci yang dapat membantu dalam merawat tanaman yang dimiliki pemain. Kurcaci 1 hanya dapat menyiram tanaman, kurcaci 2 hanya dapat memanen tanaman, sedangkan kurcaci 3 hanya dapat membersihkan tanaman yang mati. Tanaman yang dipanen oleh kurcaci akan secara otomatis dijual. Kurcaci bergerak secara cerdas menuju tanaman sesuai dengan pekerjaan yang dapat dilakukannya.

Kurcaci dapat diaktifkan dengan menggunakan perintah :

wake\_up <nomor\_kurcaci> #

selain itu kurcaci juga dapat dinon-aktifkan dengan menggunakan perintah :

sleep <nomor\_kurcaci> #

status tiap-tiap kurcaci dapat dilihat dengan menggunakan perintah :

status kurcaci #

perintah tersebut akan menampilkan status ketiga kurcaci.

Setelah pemain puas berada di lahan, maka pemain dapat pindah ke area Toko. Pemain dapat pindah ke area Toko menggunakan perintah teleport <nama\_area> #. Di area Toko, pemain dapat membeli atau menjual bibit tanaman. Untuk menjual tanaman, digunakan perintah sell <nomer\_slot> <jumlah> #. Untuk membeli tanaman, digunakan perintah buy <nomer\_slot> <jumlah> #. Untuk membeli item di Toko, maka pemain dapat melihat daftar tanaman yang dijual menggunakan perintah store #.

Setelah pemain puas melakukan transaksi di Toko, maka pemain dapat pindah ke area Lahan menggunakan perintah teleport <nama\_area> #. Setelah pemain berada di area Lahan, pemain dapat melanjutkan berbagai aktivitas yang dapat dilakukan di lahan. Jika pemain mengalami kehabisan energi, maka pemain harus melakukan perintah teleport rumah # karena rumah merupakan tempat bagi pemain untuk me-restore energinya. Jika pemain sudah merasa bosan bermain, maka selanjutnya dapat digunakan perintah exit # untuk keluar dari permainan.

Di dalam game kami ini, terdapat beberapa musim. Pada musim tertentu, maka hanya tanaman tertentu saja yang dapat tumbuh. Dan tanaman lain otomatis mati. Setelah beberapa tanaman mati akibat musim tertentu, maka selanjutnya merupakan tugas dari pemain dan kurcaci untuk melakukan aktivitas slash di lahan. Setelah tanaman yang mati dislash, maka selanjutnya lahan dapat dicangkul lagi, lalu ditanami dan disiram.

Di game kami ini juga terdapat beberapa kejadian alam yang terjadi secara acak. Kejadian alam pertama yang mungkin terjadi adalah hujan. Akibat dari hujan ini adalah seluruh lahan disiram. Kejadian alam kedua yang mungkin terjadi adalah tornado. Tornado terjadi di lahan. Akibat dari tornando ini adalah seluruh tanaman akan mati. Tornado ini hanya terjadi pada saat pergantian hari. Karena pergantian hari hanya terjadi saat pemain berada di rumah, maka player tidak mungkin terkena tornado. Kejadian alam terakhir yang terjadi adalah shining. Akibatnya yang ditimbulkan tidak ada.

# 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) |

## 

|  |  |
| --- | --- |
| Tanggal : Jum’at, 18 Maret 2011 | Catatan Rapat :  Membahas bagian rancangan laporan, isi laporan, dan pendahuluan laporan  Melakukan demo untuk prototype yang dikumpul  Interfacenya dibagusi  Check Prototype |
| Tempat : Selasar Programming (Asistensi III) |
| 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