Kelompok : 1. Micky Yudi Utama / 13514011

2. Elvina R. K. Situmorang / 13514045

3. Robert Sebastian Herlim / 13514061

4. Atika Azzahra / 13514077

Ass Mentor : Khoirunnisa Afifah

# Deskripsi Umum Aplikasi

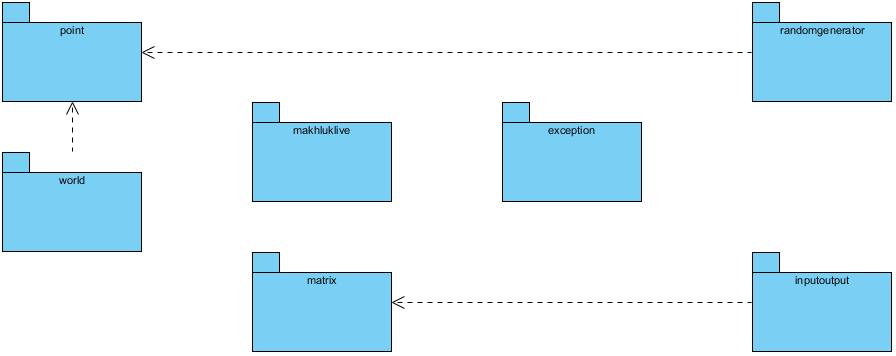
Aplikasi ini terdiri dari 2 bagian *main* *program* yaitu Animal Village dan Snake. Animal Village merupakan progam untuk menjalankan simulasi dunia. Ukuran dunia ditentukan pada awal program akan di *run* beserta dengan inisialisasi makhluk-makhluk yang ada pada dunia. Sambil berjalannya program, *user* dapat menambahkan makhluk baru lainnya juga. Program Animal Village akan berhenti ketika sudah tidak ada lagi makhluk yang berada pada dunia yang terbentuk.

Sedangkan Snake merupakan sebuah program permainan. Awal dari program ini, *user* akan memilih tingkat kesulitan permainan. Permainan ini merupakan permainan untuk menjalankan ular untuk mencari makan. Jika ular memakan makanan yang tersedia, maka panjang ular akan bertambah panjang. Tetapi, jika ular berada pada satu titik dengan pemangsanya, maka permainan berakhir. Selain itu, permainan juga dapat berakhir ketika ular menabrak pagar atau berusaha untuk keluar dunia. Makanan dan pemangsa dari ular selama permainan dimunculkan secara *random* oleh program aplikasi.

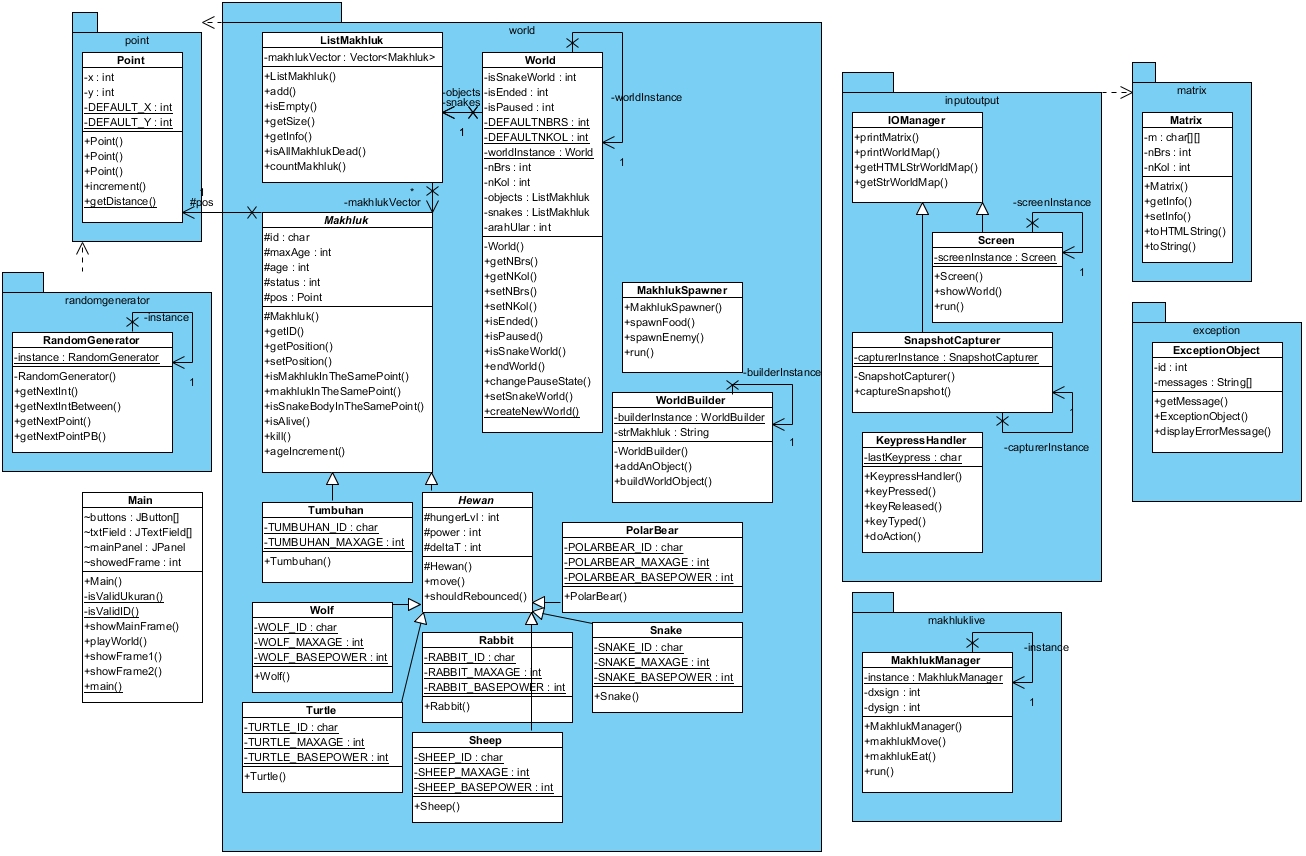
# Deskripsi Desain Teknis Aplikasi

Program ini mengimplementasikan tiga JAVA *Design Pattern* yaitu *Singleton*, *Builder*, dan MVC. *Design Pattern Singleton* diimplementasikan kepada kelas-kelas yang memang tidak diperlukan penciptaan baru lagi saat aplikasi berjalan, seperti *World, Screen, SnapshotCapturer, RandomGenerator,* dan *WorldBuilder*. *Design Pattern Builder* diimplementasikan oleh kelas *WorldBuilder* yang merupakan kelas untuk membangun kelas *World* dan bertanggung jawab untuk menginisialisasi objek-objek dunia yang direpresentasikan oleh kelas *Makhluk* seperti *PolarBear, Rabbit, Sheep, Snake, Tumbuhan, Turtle,* dan *Wolf*. *Design Pattern* MVC diimplementasikan oleh kelas *World* sebagai model, kelas *Screen* dan *SnapshotCapturer* sebagai View, dan kelas *KeypressHandler, MakhlukManager,* dan *MakhlukSpawner* sebagai *Controller* objek-objek dunia.

# Diagram Paket



# Diagram Kelas



## Ulasan Redesign

Desain kelas dari versi CPP ke versi JAVA mengalami beberapa perubahan. Hal ini dikarenakan dibentuk sebuah permainan bernama *“Snake”.* Untuk membentuk game *“Snake”* ini, dibentuk kelas baru bernama kelas *MakhlukSpawner*. Kelas ini berfungsi untuk menambahkan *instance* makhluk (kecuali ular dan tumbuhan) secara berkala. Selain itu versi JAVA menggunakan *thread* sehingga kelas *MakhlukSpawner, Screen*, dan *MakhukManager* jadi mengimplemenetasikan *interface Runnable* untuk meng-*override* method *run*. GUI untuk program ini diimplementasikan menggunakan *JAVA Swing* yang berada di kelas *Main.*

# Penggunaan Ulang Kelas dari Tugas-tugas yang lalu

## Penggunaan Ulang kelas CPP

| **No** | **Nama Kelas CPP** | **Nama Method** | **Deskripsi Perubahan** |
| --- | --- | --- | --- |
| 1 | Makhluk | getID  getAge  getMaxAge  getPosition  setPosition  isMakhlukInTheSamePoint  makhlukInTheSamePoint  isAlive  kill  ageIncrement | - |
| 2 | Hewan | move  getPower  shouldRebounced | - |
| 3 | PolarBear | - | - |
| 4 | Rabbit | - | - |
| 5 | Turtle | - | - |
| 6 | Snake | - | - |
| 7 | Wolf | - | - |
| 8 | Sheep | - | - |
| 9 | Tumbuhan | - | - |
| 10 | Point | getX  getY  setX  setY  increment  getDistance | - |
| 11 | Matrix | getNBrs  getNKol  getInfo  setInfo | - |
| 12 | RandomGenerator | getInstance  getNextInt  getNextIntBetween  getNextPoint  getNextPointPB | - |
| 13 | LMakhluk | add  isEmpty  getSize  getInfo  isAllMakhlukDead | - |
| 14 | WorldBuilder | getBuilderInstance  getStrMakhluk  setStrMakhluk  addAnObject  buildWordlObject | - |
| 15 | World | getObjects  getWorldInstance  getNBrs  getNKol  setNBrs  setNKol  isEnded  isPaused  endWorld  changePauseState | - |
| 16 | MakhlukLive | getInstance  makhlukMove  makhlukEat | - |
| 17 | IOManager | printMatrix  printWorldMap | - |
| 18 | Screen | getScreenInstance  showWorld | - |
| 19 | SnapshotCapturer | getCapturerInstance  saveOldBuf  resetCoutBuf  captureSnapshot | - |
| 20 | KeypressHandler | getHandlerInstance  getLastKeypress  getKeypress  doAction | - |
| 21 | ExceptionObject | displayErrorMessage | - |

## Penggunaan Ulang kelas Java

| **No** | **Package** | **Nama Kelas JAVA** | **Nama Method JAVA** | **Deskripsi Perubahan** |
| --- | --- | --- | --- | --- |
| 1 | world | Makhluk  Hewan  PolarBear  Rabbit  Sheep  Snake  Tumbuhan  Turtle  Wolf  ListMakhluk  MakhlukSpawner  World  WorldBuilder | getID  getAge  getMaxAge  getPosition  setPosition  setPosition  isMakhlukInTheSamePoint  makhlukInTheSamePoint  isSanekBodyInTheSamePoint  isAlive  kill  ageIncrement  move  getPower  shouldRebounced  -  -  -  -  -  -  -  add  isEmpty  getSize  getInfo  isAllMakhlukDead  countMakhluk  spawnFood  spawnEnemy  run  getObjects  getSnakes  getWorldInstance  getNBrs  getNKol  setNBrs  setNKol  getArahUlar  isEnded  isPaused  isSnakeWorld  endWorld  changePauseState  setSnakeWorld  setArahUlar  getBuilderInstance  getStrMakhluk  setStrMakhluk  addAnObject  buildWorldObject | Pada kelas makhluk ditambahkan method isSnakeBodyInTheSamePoint untuk menangani program Snake untuk memberitahu apakah kepala ular dan badannya ada pada tempat yang sama atau tidak. Method ini akan digunakan untuk menyelesaikan permainan jika kepala ular berada pada tempat yang sama dengan badannya.  -  -  -  -  -  -  -  -  Kelas yang awalnya bernama LMakhluk diubah menjadi ListMakhluk atas dasar *clean code*.  Method countMakhluk ditambahkan untuk menghitung makhluk. Makhluk ini digunakan untuk memudahkan perhitungan jumlah makanan dan pemangsa yang ada di dalam dunia permainan.  Kelas ini ditambahkan untuk memenuhi kebutuhan dalam pembuatan permainan Snake. spawnFood berfungsi untuk menambahkan makanan bagi ular sedangkan spawnEnemy untuk menambahkan pemangsa bagi ular.  Run yang merupakan method overiding java.lang.Runnable.run berfungsi untuk menjalankan spawnFood dan spawnEnemy  getSnakes berguna untuk mengembalikan list of ular yang ada di world  getArahUlar mengembalikan arah hadap ular  isSnakeWorld mengembalikan apakah world (dunia/program) yang sedang dijalankan adalah dunia/permainan Snake atau tidak  setSnakeWorld berguna untuk mengubah state World biasa (Animal Village) ke state permainan Snake  setArahUlar berguna untuk mengubah state arah hadap ular  - |
| 2 | point | Point | getX  getY  setX  setY  increment  getDistance | - |
| 3 | matrix | Matrix | getnBrs  getnKol  getInfo  setInfo  toHTMLString  toString | Sebagian besar re-use, perubahan karena penanganan I/O berbeda pada JAVA |
| 4 | randomgenerator | RandomGenerator | getNextInt  getNextIntBetween  getNextPoint  getNextPointPB | - |
| 5 | makhluklive | MakhlukManager | getInstance  makkhlukMove  makhlukEat  run | Perubahan nama dari MakhlukLive menjadi MakhlukManager supaya sesuai dengan aturan penamaan, yaitu menggunakan *noun*.  Penambahan method run yang digunakan saat penciptaan sebuah thread |
| 6 | inputoutput | IOManager  Screen  SnapshotCapturer | printMatrix  printWorldMap  getHTMLStrWorldMap  getStrWorldMap  getScreenInstance  showWorld  run  getCapturerInstance  captureSnapshot | Perubahan karena fitur pure-JAVA yang tidak se-*wideapplicable library* CPP. Tetapi tidak ada perubahan dengan implementasi JAVA pada TB2  Penambahan methd run untuk menangani thread pada JAVA  Tidak memungkinkan untuk *redirect cout buffer* di JAVA (perubahan ini sudah dari TB2) |
| 7 | exception | ExceptionObject | getMessage  displayErrorMessage | getMessage digunakan untuk melakukan JUnit tes. |

# Pengukuran Metriks Aplikasi

Pengukuran metriks aplikasi digunakan dengan menggunakan *tools* JDepend. JDepend adalah sebuah *tools* yang digunakan untuk mengukur metriks yang menentukan kualitas rancangan dengan menganalisa kelas-kelas Java yang ada. Hasil yang didapat dari *tools* ini menentukan tingkat *extensibility, reusability,* dan *maintainability* dari program yang sudah dirancang.

Nilai metrik yang diberikan oleh *tools* JDepend berupa:

1. CC (*Concrete Class Count*)merupakan jumlah kelas tidak abstrak dalam sebuah *package*
2. AC (*Abstract Class Count*) merupakan jumlah kelas abstrak dalam sebuah *package*
3. CA (*Afferent Couplings*) merupakan jumlah *package* yang menggunakan sebuah *package*
4. CE (*Efferent Couplings*) merupakan jumlah *package* yang digunakan oleh sebuah *package*
5. A (*Abstractness*) merupakan rasio kelas abstrak dan *interface.* Nilai A=0 menunjukkan sebuah *package* sepenuhnya konkret dan nilai A=1 menunjukkan sebuah *package* sepenuhnya abstrak
6. I (*Instability*) merupakan tingkat kestabilan sebuah *package*. Nilai I=0 menunjukkan *package* sepenuhnya stabil dan nilai I=1 menunjukkan *package* sepenuhnya tidak stabil
7. D (*Distance from The Main Sequence*) merupakan tingkat keseimbangan antara abstraksi dan stabilitas sebuah *package*. Sebuah rancangan yang ideal memiliki nilai D=0 dengan kriteria sebagai berikut:

* Rancangan yang sangat stabil (I=0) dan sangat abstrak (A=1)
* Rancangan yang tidak stabil (I=1) dan sangat konkret (A=0)

1. V (*Volatility*) merupakan nilai yang diatur melalui jdepend.properties. Nilai V=0 menunjukkan sebuah *package* tidak akan pernah diubah dan nilai V=1 menunjukkan bahwa *package* dapat berubah.

Pada hasil perhitungan metriks, nilai yang paling penting merupakan nilai D yaitu jarak dari keidealan. Nilai D yang semakin mendekati nol menunjukkan semakin ideal sebuah *package*. Berikut merupakan ringkasan nilai D dari program kami:

1. *Package* exception (D=0,6)
2. *Package* inputoutput (D=0,09)
3. *Package* makhluklive (D=0,29)
4. *Package* matriks (D=0,5)
5. *Package* point (D=0,8)
6. *Package* randomgenerator (D=0,4)
7. *Package* world (D=0,22)

Secara keseluruhan desain yang kami rancang masuk dalam kategori bagus melihat sebagian besar nilai D pada *package* lebih mendekati 0. Selain itu tidak terdapat *Cyclic* pada hasil rancangan kami.

# Pengukuran Kualitas Kode Aplikasi

*CheckStyle* yang digunakan pada program ini adalah *sun style default*. Dari 31 file yang dicek menggunakan *CheckStyle* terdapat 5 file yang mendapatkan peringatan *error*. Berikut adalah nama file program yang mendapatkan peringatan dan detail *error-*nya*.* Hasil *CheckStyle* dalam file berkestansi xml tercantum pada bagian *Lampiran*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nama File** | **Jumlah Error** | **Rule** | **Kategori** |
| SnapshotCapturer.java | 2 | JavadocMethod | Javadoc |
| KeypressHandler.java | 1 | JavadocMethod | Javadoc |
| Hewan.java | 3 | VisibilityModifier | Design |
| Makhluk.java | 5 | VisibilityModifier | Desgin |
| Main.java | 66 | Magic Number | Coding |
| JavadocMethod | Javadoc |

Selain menggunakan *Checkstyle,* kami juga menggunakan *tools* PMD untuk pengukuran kualitas kode aplikasi. Beberapa hal yang dapat dideteksi oleh PMD yaitu:

1. Bug yang mungkin terjadi (blok Etty/catch/finally/switch yang kosong)
2. Kode mati (variabel, parameter, ataupun method yang tidak terpakai)
3. Kondisi if/while yang kosong
4. Ekspresi yang terlalu rumit
5. Kode suboptimal (penggunaan kode yang tidak perlu)
6. Kelas dengan ukuran *Cyclomatic Complexity* yang tinggi
7. Dupilkasi kode

Pada program yang kami buat. Sudat tidak terdapat eror yang dideteksi oleh *tools* PMD.

# Dokumentasi Rinci Kelas

Disetor langsung ke *Oddyseus* dalam bentuk *softcopy*.