**GAME TEBAK NAMA LATIN BUAH BEBASIS ANDROID**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Muamal Khoerudin  1127050099  Teknik Informatika  UIN Sunan Gunung Djati Bandung  Muamal.khoerudin@gmail.com | Didi Sonhaji  1127050039  Teknik Informatika  UIN Sunan Gunung Djati Bandung | Agung Ramadhan  1127050003  Teknik Informatika  UIN Sunan Gunung Djati Bandung |  |  |

*Abstrak:* Pembelajaran terhadap anak dapat dilakukan dengan berbagai cara, salah satunya yaitu denganmenggunakan teknologiyang sedang berkembang yaitu *mobile.*saat ini banyak orang yang membangunakan aplikasi untuk membantu dalam hal pembelajaran yang menarik dan praktis yang dapat diakses menggunakan *mobile, g*ame tebak nama latin buah ini merupakan salah satunya. Game ini meruapakan aplikasi yang mengenalkan nama latin dari buah-buahan berserta gambarnya. Dengan adanya game ini diharapakan dapat memberikan informasi mengenai nama latin pada setiap buah.

|  |
| --- |
|  |

**2. LANDASAN TEORI**

**2.1 Android**

Android adalah sistem operasi berbasis Linux yang dipergunakan sebagai pengelola sumber daya perangkat keras, baik untuk ponsel, smartphone dan juga PC tablet. Secara umum Android adalah platform yang terbuka (Open Source) bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai piranti bergerak. Menurut Wikipedia *Android* adalah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia.

**a. SDK**

Android-SDK merupakan tools bagi para programmer yang ingin mengembangkan aplikasi berbasis google android. Android SDK mencakup seperangkat alat pengembangan yang komprehensif. Android SDK terdiri dari debugger, libraries, handset emulator,dokumentasi, contoh kode, dan tutorial. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk desktop modern), Mac OS X 10.4.8 atau lebih, Windows XP atau Vista. Persyaratan mencakup JDK, Apache Ant dan Python 2.2 atau yang lebih baru. IDE yang didukung secara resmi adalah Eclipse 3.2 atau lebih dengan menggunakan pluginAndroid Development Tools (ADT), dengan ini pengembang dapat menggunakan teks editor untuk mengedit file Java dan XML serta menggunakan peralatan command line untuk menciptakan, membangun, melakukan debug aplikasi Android dan pengendalian perangkat Android (misalnya, reboot, menginstal paket perangkat lunak dengan jarak jauh).

**b. ADT**

Android Development Tools (ADT) adalah plugin untuk Eclipse yang didesain untuk pengembangan aplikasi Android. ADT memungkinkan Eclipse untuk digunakan dalam membuat aplikasi Android baru, membuat User Interface, menambahkan komponen berdasarkan framework API Android, debug aplikasi, dan pemaketan aplikasi Android.

Android Development Tools (ADT) adalah plugin yang didesain untuk IDE Eclipse yang memberikan kita kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk Eclipse akan memudahkan kita dalam membuat aplikasi project android, membuat GUI aplikasi, dan menambakan komponen-kompenen yang lainnya, begitu juga kita dapat melakukan running aplikasi menggunakan Android SDK melalui Eclipse. Dengan ADT juga kita dapat melakukan pembuatan package android (.apk) yang digunakan untuk distribusi aplikasi android yang kita rancang.

**c. Android Studio**

Android Studio merupakan lingkungan pengembangan Android baru berdasarkan IntelliJ IDEA. Mirip dengan Eclipse dengan ADT Plugin, Android Studio menyediakan alat pengembang terintegrasi untuk pengembangan dan debugging.

Android Studio menawarkan:

1. Berbasis Gradle.
2. Android-spesifik refactoring dan perbaikan yang cepat.
3. Alat Lint untuk menangkap kinerja, kegunaan, versi kompatibilitas dan masalah lainnya.
4. ProGuard dan app-signature.
5. Wizard untuk design dan membuat komponen-komponen umum Sebuah layout editor yang memungkinkan untuk drag-and-drop komponen UI, pratinjau layout pada beberapa konfigurasi layar, dan banyak lagi.
6. Built-in dukungan untuk Google Cloud platform, sehingga mudah untuk mengintegrasikan Google Cloud Messaging dan App Engine sebagai komponen server-side.

**2.2 UML *(Unified Modeling Language)***

UML *(Unified Modeling Language)* merupakan pengganti dari metode analisis berorientasi object dan design berorientasi object (O OA&D) yang dimunculkan sekitar akhir tahun 80-an dan awal tahun 90-an.

UML merupakan gabungan dari metode Booch, Rumbaugh (OMT) dan Jacobson. Tetapi UML ini akan mencakup lebih luas daripada OOA&D. Pada pertengahan pengembangan UML dilakukan standarisasi proses dengan OMG (Object Management Group) dengan harapan UML akan menjadi bahasa standar pemodelan pada masa yang akan datang.

UML disebut sebagai bahasa pemodelan bukan metode. Kebanyakan metode terdiri paling sedikit prinsip, bahasa pemodelan dan proses. Bahasa pemodelan (sebagian besar grafik) merupakan notasi dari metode yang digunakan untuk mendesain secaracepat.

Bahasa pemodelan merupakan bagian terpenting dari metode. Ini merupakan bagian kunci tertentu untuk komunikasi. Jika anda ingin berdiskusi tentang desain dengan seseorang, maka Anda hanya membutuhkan bahasa pemodelan bukan proses yang digunakan untuk mendapatkan desain.

UML merupakan bahasa standar untuk penulisan blueprint software yang digunakan untuk visualisasi, spesifikasi, pembentukan dan pendokumentasian alat-alat dari sistem perangkat lunak.

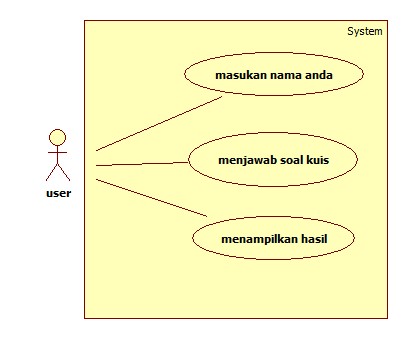
**3. PERANCANGAN PERANGKAT LUNAK**

**3.1 *Use Case***

Use cas diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu.

Sebuah use case dapat meng-include fungsionalitas use case lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa use case yang di-include akan dipanggil setiap kali use case yang meng-include dieksekusi secara normal. Sebuah use case dapat di-include oleh lebih dari satu use case lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common.

Sebuah use case juga dapat meng-extend use case lain dengan behaviour-nya sendiri. Sementara hubungan generalisasi antar use case menunjukkan bahwa use case yang satu merupakan spesialisasi dari yang lain.



**3.2 *Class Diagram***

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok:

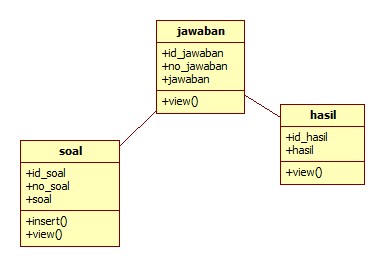
1. Nama (dan stereotype)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

* *Private*, tidak dapat dipanggil dari luar class yang bersangkutan
* *Protected*, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
* *Public*, dapat dipanggil oleh siapa saja

Hubungan Antar Class

1. *Asosiasi*, yaitu hubungan statis antar class. Umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain. Panah *navigability* menunjukkan arah query antar class.
2. *Agregasi*, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. *Pewarisan*, yaitu hubungan hirarkis antar class. Class dapat diturunkan dari class lain dan mewarisi semua atribut dan metoda class asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari class yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. *Hubungan dinamis*, yaitu rangkaian pesan (message) yang di-passing dari satu class kepada class lain. Hubungan dinamis dapat digambarkan dengan menggunakan sequence diagram yang akan dijelaskan kemudian.



**3.3 *Sequence Diagram***

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan. Masing-masing objek, termasuk aktor, memiliki lifeline vertikal. Message digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, message akan dipetakan menjadi operasi/metoda dari class.

Activation barmenunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan icon khusus untuk objek boundary, controller dan persistent entity.

**4. IMPLEMENTASI**

**5. PENUTUP**