Dokumen ini berisi pedoman serahan Tugas Besar 1 (dengan tema Virtual Zoo)
Tujuan dokumen ini ditulis adalah agar mahasiswa maupun penilai mempunyai kriteria yang sama terhadap artefak yang harus diserahkan, dan kriteria kualitasnya

BAGIAN I : PEDOMAN

1 Nama Kelompok Pembuat Tugas

1. Nama kelompok sebaiknya mempunya arti dan serius sebagai produk, bukan hanya main-main, dan dapat anda jelaskan kenapa nama tersebut dipilih
2. Memberikan konotasi positif
3. Jika singkatan, dijelaskan artinya
4. Bukan kalimat aneh-aneh

2 Source Code

1. Source code disimpan dengan rapi sesuai dengan kaidah pembuatan direktori/sub direktori. Setiap kelas satu direktori. Aplikasi menjadi root direktori
2. Dokumentasi source code dibuat dengan menggunakan Doxygen
    a. Tambahkan anotasi pada source code agar dokumentasi kelas dapat diperoleh. Deskripsi Kelas harus mencerminkan *Responsibility*nya
    b. Hasil pembangkitan dokumentasi dengan Doxygen disimpan dalam bentuk softcopy shingga semua kelas, atribut dan method jelas. Setiap Method terdokumentasi dengan jelas : I.S/Prekondisi, Deskripsi ringkas, F.S/PostCond

3 Dokumentasi Perangkat Lunak

1. Dokumentasi yang anda serahkan dibuat dengan templates bebas, namun harus mengandung elemen-elemen yang diberikan dalam kerangka Dokumentasi pada lampiran (ada yang harus diprint dan diserahkan dalam bentuk hardcopy, semuanya harus diserahkan dalam bentuk softcopy dan diupload di Olympia.
2. Sesuai dengan kerangka yang diberikan (lihat Lampiran)
3. Deskripsi Umum aplikasi ditulis dalam kalimat bahasa Indonesia/Inggris
    a. Jelas “konteks”nya
    b. Diuraikan dalam kalimat yang ringkas dan jelas
4. Daftar Kebutuhan Fungsional
    a. Dituliskan dalam bentuk tabel, dengan identifikasi/penomoran fungsi serta deskripsi yang jelas
    b. Penomoran fungsional didasari oleh **fitur aplikasi dari kaca mata pengguna**, **bukan per method**
    c. Setiap kebutuhan fungsional akan mempunyai satu atau lebih skenario, yang akan didekomposisi menjadi satu atau lebih butir uji (test case)
5. Rancangan seknario dan “Test Case” (kasus normal, dan kasus alternatif yang harus ditest)
    a. Harus runut terhadap Kebutuhan Fungsional
    b. Akan dipakai sebagai dasar melakukan Demo.
    c. Saat membuat test case, anda harus membayangkan sedang melakukan demo. Urutan demo sesuai test case

6. Diagram kelas
    a. Daftar Kelas dan *traceability* dengan versi
    b. Gambar diagram kelas dengan *tools* dan penjelasannya kenapa rancangannya demikian. Jika memakai design pattern, jelaskan design pattern yang dipakai
    c. Penjelasan setiap kelas mengacu ke dokumentasi source code dengan Doxygen yang anda jadikan lampiran (penjelasan tidak perlu diketik ulang, karena sudah di-generate dengan Doxygen, lihat diatas)
    d. *Responsibility*nya
    e. Hubungannya dengan kelas lain (*dependency*nya)
    f. Gambaran umum atribut dan methodnya
    g. Penamaan kelas sesuai dengan kaidah yang dijelaskan pada butir berikut ini

4 Penamaan Kelas

1. Nama Kelas harus kata benda, atau Verbal Noun (kata benda yang berasal dari kata benda)
    a. Jika nama kelas adalah kata benda, akan menjadi ADT, yaitu boleh mempunyai method sesuai operasi yang dapat dilakukan oleh "type" data tsb
    b. Jika nama kelas adalah verbal noun (misalnya Sender, Receiver, Manager, Converter, Controller,..) maka nama kelas harus sesuai dengan responsibility-nya
2. Nama kelas harus menunjukkan *responsibility*nya
3. Deskripsi kelas harus mencerminkan kategorinya : ADT, Mesin, Proses
4. Deskripsi kelas harus mencakup atribut/state dan behaviournya atau fungsi yang diembannya
5. Jika ada *inheritance*, nama kelas anak harus kompatibel dengan bapak (hubungan *Is-A*)
6. Jika bukan utilities, sebaiknya ada satu kelas bernama Main
7. Penulisan nama kelas JAVA harus kompatibel dengan aturan kode JAVA
8. Pada tahap rancangan kelas, hanya munculkan atribut dan method yang akan mencerminkan *service* atau *behaviour* kelas. 4 sekawan (ctor, cctor, dtor, operator=) serta *getter* dan *setter* tidak perlu dituliskan.
9. Method :
    a. Nama Method harus kata kerja
    b. Nama predikat harus mengembalikan boolean dan menunjukkan konotasi positif bukan negasi, misalnya IsFound() dan bukan NotFound().
    c. Parameter method harus diperhatikan apakah hanya mengubah "`this`" (dirinya). Parameter objek dari kelas lain harus dicermati apakah memang layak (kecuali koverter objek lain ke dirinya).

5 Penamaan Package

1. Nama Package harus kata benda
2. Pemaketan harus jelas kriteria pengelompokan kelas di dalamnya. Jelaskan kriteria pemaketan sehingga diperoleh packages yang disajikan
3. Kelas yang menjadi isi package harus sesuai dengan kriteria pengelompokan

6 Rancangan Software

1. Harus dibuktikan minimal S.O.L.I.D.
2. Harus dapat dijelaskan pada top level design
3. Jika mungkin, diajukan beberapa alternatif dan ada analisis pemilihan rancangan yang menjadi dasar pemilihan versi yang diimplementasi

7 Metriks Perangkat Lunak

1. Data diperoleh dari struktur direktori dan nama file, dengan root direktori nama aplikasi. Anda dapat melakukan dengan perintah Unix (bukan mengetik satu per satu)
2. Anda hanya perlu mengisi mengcopy teks sbb dan memberikan angka pada kolom "Besarnya". Anda tidak perlu menggunakan tools untuk menghitung metriks perangkat lunak tugas Zoo Animal ini.
3. Dari data metriks yang anda sajikan, anda harus dapat menyimpulkan kualitas, besaran dan kompleksitas dari perangkat lunak yang dihasilkan

8 Testing Perangkat Lunak

1. Testing minimal dilakukan dalam dua tahap
    a. Unit test
    b. Functional Test
2. Test dirancang sesuai dengan skenario yang mencakup semua test case (test coverage cukup)

9 Teks Dokumentasi dan Laporan lainnya

1. Kalimat harus lengkap dan untuk dokumen teknis, harus SPO (ada Subjek yang jelas)
2. Bahasa Spesifikasi harus ringkas, jelas, presisi dan sesuai pola. Misalnya :
    a. Deskripsi fungsional (Prekondisi,definisi, spesifikasi)
    b. Deskripsi proses (I.S ~Prekondisi., F.S~post condition, deskripsi Proses)
    c. Deskripsi perilaku (rangkaian proses akibat kolaborasi atau kirim mengirim pesan)
    d. Deskripsi struktur (kelas, objek, ...) – hubungan antar kelas dan deskripsi "sekeluarga" kelas-kelas
3. Satu paragraf berisi satu ide
4. Spesifikasi tingkat paling atas (top level) beberapa baris yang mencerminkan perilaku utama.
5. Beberapa kata yang tidak sesuai untuk membuat spesifikasi "*software*":
    a. Memasukkan {untuk mendeskripsi input}
    b. Mengeluarkan {untuk mendeskripsi output}
    c. Inputan {seharusnya input}
6. Kata harus ditulis sesuai ejaan bahasa Indonesia dan dipakai sesuai artinya, beberapa kata yang sering dipakai tak sesuai atau salah ketik
    a. Didalam {kata depan harus dipisah} seharusnya di dalam
    b. Merubah {salah sintaks bahasa Indonesia} seharusnya mengubah
    c. Memiliki {pertimbangkan apakah memang tak dapat diganti sesuai semantik/artinya}
    d. Merupakan {pertimbangkan apakah sebetulnya berarti "adalah" }
    e. Sintaks : artinya aturan tata bahasa

7. Format dokumen
    a. Font untuk teks harus dibedakan dengan kode program
    b. Tabel yang lebih dari satu halaman harus ada "heading" nya per halaman.
    c. Gambar harus diberi judul dan penjelasan, dan memakai konvensi yang baku (diagram kelas, DFD, Flowchart, ...)
    d. Style "bullet" hanya untuk PPT, sebaiknya tidak dipakai untuk dokumen karena teks dalam dokumen sebaiknya bisa diacu sesuai dengan nomornya
8. Gambar dan Tabel
    a. Sebaiknya diberi nomor dan judul
    b. Tabel harus diulang judulnya (*heading repeated*)
    c. Gambar harus dapat dibaca (perhatikan ukurannya)

10 Log Activity

1. Log activity dibuat sesuai dengan apa yang dikerjakan, dan dibuat secara bertahap
2. Jelas pembagia perannya dan dilakukan dengan baik
3. Memanfaatkan waktu dengan efektif untuk mencapai *deliverables*

11 Referensi Penamaan

Penamaan file *source code* harus mengikuti pedoman umum untuk bahasa C++ yang ada di Olympia. Selain itu, sebaiknya ikuti pula gaya yang tertera pada pedoman tersebut.

Bagian II: KERANGKA DAN FORM

Bagian ini hanya berisi kerangka (*templates*) dari bagian-bagian dokumentasi yang harus diserahkan. Kriteria kualitas isinya, yang akan mencerminkan nilai anda, dituliskan dalam dokumen lain

Kerangka Dokumentasi Utama Virtual Zoo Versi XX.XX [dibuat per versi dan dicetak]

1. Cover (identitas kelompok, judul,..)
2. Deskripsi umum Aplikasi
3. Rancangan Kelas
   - 3.1. Ulasan Ringkas Rancangan Kelas (mengacu ke gambar dan penjelasan pada 2.2., 2.3,… dst). Jika merupakan kode hasil evolusi/perubahan versi sebelumnya, dilengkapi dengan ulasan perubahan yang dilakukan. Anda juga harus menjelaskan perbedaan rancanga anda dengan diagram kelas terakhir yang dihasilkan dari proses reverse engineering (jika ada perbedaan) atau dengan hasil Doxygen
   - 3.2. Rancangan-1
   - 3.3. Rancangan-2
   - 3.4. (dst…. sebanyak rancangan yang anda buat. Sebaiknya setiap konfigurasi rancangan diberi identifikasi)
4. *Snapshot* struktur direktori/sub direktori [diprint]
5. Daftar Lampiran (perhatikan mana yang dicetak daftarnya saja, dan mana yang lampiran yang tidak perlu dicetak, hanya di-*upload* dalam bentuk softcopy)
   - 5.1. Skenario Test [dicetak dan dilampirkan]
   - 5.2. Penghitungan Metriks Perangkat Lunak [dicetak dan dilampirkan]
   - 5.3. Log activity [dicetak dan dilampirkan]
   - 5.4. Diagram kelas hasil *reverse engineering* source code [softcopy]
   - 5.5. Hasil pembangkitan dokumentasi dengan doxygen [softcopy]
   - 5.6. Hasil Unit Test dengan GoogleTest [softcopy]
   - 5.7. Hasil Static Code Test dengan CppCheck [softcopy]

Kerangka Log Activity [dibuat dan dicetak hanya 1 untuk seluruh versi Tugas besar)

1. Pembagian Peran
   - 1.1. Deskripsi secara Umum
   - 1.2. Deskripsi rinci tanggung jawab, minimal berisi tabel sbb. Yang menunjukkan tanggung jawab terhadap setiap unit

Kelas	**File**	**Boleh diisi lebih dari Satu**	
		Designer	**Implementor/Koder**

Dokumentasi : jelaskan sesuai pembagian tugas dan dekomposisi dokumen yang ditentukan

Elemen Dokumentasi	**Writer**	**Reviewer**

2. Rincian kegiatan, minimal berisi tabel sbb

No	**Dari Tgl… Pk…**	**S.d. Tgl. .. Pk…**	**Kegiatan**	**Hasil**	**Keterangan**
1.					
2.					
3.					
4.					
5.					
6.					
7.					

Total waktu yang dibutuhkan untuk mengerjakan tugas : …. manhours

3. Rangkuman capaian

Jelaskan rangkuman capaian tugas anda berdasarkan fakta-fakta yang anda ungkapkan.

Kerangka Skenario Test

Bagian ini berisi kerangka dan sekaligus lembar penilaian Testing. Akan diacu saat demo

Form Penilaian Functional Test

Functional test dilakukan terhadap setiap versi

Versi : ……

(1 baris adalah 1 butir test)

Skenario	Keterangan	Fakta	Nilai Mhs	Nilai Ass
General				
Inisiasi, View kemudian quit				
Display Virtual Zoo				
Tour Virtual Zoo				
Bonus				
Save ke file				
Retrieve dari file				

Keterangan dan Fakta diisi oleh mahasiswa dengan OK/tidak
Nilai akan diisi oleh mahasiswa/asisten dengan nilai abjad (A, B, C, E)

Form penilaian Unit Test

Bagian ini berisi daftar Unit Test yang sudah anda lakukan terhadap kelas atau sekumpulan kelas yang tak mungkin untuk ditest sendiri secara unit test (misalnya kelas dengan hubungan inheritance, atau kelas abstrak)
Anda hanya mengisi OK/NO (tidak OK). Nilai akan diisi oleh asisten
(1 baris adalah 1 butir test; 1 butir Test adalah satu butir kasus uji)

Kelas/ keluarga kelas	**Method**	**Kasus**	**OK/NO**	**Nilai**
XXX	M1	Case-1		
		Case-2		

Nilai : ……

Lembar Penilaian Metriks PL

Bagian ini harus diisi dan dicetak, sekaligus merupakan lembar penilaian

Versi : VZ01 – pada kondisi terakhir

No	Metriks	Besarnya	Keterangan
1.	Number of packages		Package adalah sub direktori anda
2.	Number of Classes		Jumlah kelas riil
3.	Number of Abstract class		Pada CPP interface adalah ABC
4.	Afferent Couplings (Ca):		Lihat definisi di bawah
5.	Efferent Coupling (Ce)		Lihat definisi di bawah
6.	Asbtractness (A)		Rasio jumlah kelas abstract dibandingkan kelas riil
7.	Instability (I)		Lihat rumus di bawah
8.	Package Dependency Cycle	Ada/Tidak	Adakah dependency Cycle? Artinya paket yang saling tergantung ? seharusnya tidak boleh ada karena tidak bisa dibuilt
9.	Kelas generik		Sebutkan nama kelas generik

Versi : VZ02 – tanpa generik/inheritance

No	Metriks	Besarnya	Keterangan
1.	Number of packages		Package adalah sub direktori anda
2.	Number of Classes		Jumlah kelas riil
3.	Number of Abstract class		Pada CPP interface adalah ABC
4.	Afferent Couplings (Ca):		Lihat definisi di bawah
5.	Efferent Coupling (Ce)		Lihat definisi di bawah
6.	Asbtractness (A)		Rasio jumlah kelas abstract dibandingkan kelas riil
7.	Instability (I)		Lihat rumus di bawah
8.	Package Dependency Cycle	Ada/Tidak	Adakah dependency Cycle? Artinya paket yang saling tergantung ? seharusnya tidak boleh ada karena tidak bisa dibuilt
9.	Kelas generik		Sebutkan nama kelas generik

Versi : VZ03 dengan generik/inheritance

No	Metriks	Besarnya	Keterangan
1.	Number of packages		Package adalah sub direktori anda
2.	Number of Classes		Jumlah kelas riil
3.	Number of Abstract class		Pada CPP interface adalah ABC
4.	Afferent Couplings (Ca):		Lihat definisi di bawah
5.	Efferent Coupling (Ce)		Lihat definisi di bawah
6.	Asbtractness (A)		Rasio jumlah kelas abstract dibandingkan kelas riil

No	Metriks	Besarnya	Keterangan
7.	Instability (I)		Lihat rumus di bawah
8.	Package Dependency Cycle	Ada/Tidak	Adakah dependency Cycle? Artinya paket yang saling tergantung ? seharusnya tidak boleh ada karena tidak bisa dibuilt
9.	Kelas generik		Sebutkan nama kelas generik

Catatan : kelas generik dihitung sebagai kelas abstrak

Catatan : besaran di atas cukup anda hitung secara manual

1. Number of Classes and Interfaces: The number of concrete and abstract classes (and interfaces) in the package is an indicator of the extensibility of the package.
2. Afferent Couplings (Ca): The number of classes in other packages that depend upon classes within the package is an indicator of the package's responsibility. Afferent = incoming.
3. Efferent Couplings (Ce): The number of classes in other packages that the classes in the package depend upon is an indicator of the package's dependence on externalities. Efferent = outgoing
4. Abstractness (A): The ratio of the number of abstract classes (and interfaces) in the analyzed package to the total number of classes in the analyzed package. The range for this metric is 0 to 1, with A=0 indicating a completely concrete package and A=1 indicating a completely abstract package.
5. Instability (I): The ratio of efferent coupling (Ce) to total coupling (Ce + Ca) such that I = Ce / (Ce + Ca). This metric is an indicator of the package's resilience to change. The range for this metric is 0 to 1, with I=0 indicating a completely stable package and I=1 indicating a completely unstable package.
6. **Package Dependency Cycles:** Package dependency cycles are reported along with the hierarchical paths of packages participating in package dependency cycles.

Contoh menghitung metriks:
http://www.codemanship.co.uk/parlezuml/metrics/OO%20Design%20Principles%20&%20Metrics.pdf