UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Software Engineering
project for the final course exam

A.A. 2024-25

luca boldrin

Topics

- Regole generali
- Deliverables
- Valutazione
- Argomento del progetto

REGOLE GENERALI

- Il progetto dovrà essere realizzato in team **composto da 4 persone**. Registrare il team in: https://docs.google.com/spreadsheets/d/

  (ovviamente registrarsi anche su uniweb)
- Il team si deve registrare entro le date di consegna del progetto
- Utilizzare il linguaggio **Java** - Effettuare unit testing (utilizzando **JUnit**).
- User stories su jira
- Altra documentazione su github.
- Per i diagrammi UML si può usare un tool (plantuml, etc ....)
- Si consiglia di utilizzare IntelliJ, visualstudio o Eclipse come IDE (si può usare maven per le dipendenze)

CONSEGNA

- Alla consegna, inviare in ogni caso una **mail a luca.boldrin@unipd.it - in CC tutti i componenti del Gruppo**
- Creare Il progetto su Github (una guida). Invitare l'utente "luca-unipd" al GitHub.
- Scadenze (pending confirmation):
  - 9-11/6/25 Ore 8:30 aula Oe - presentazione progetto e iscrizione entro 1 giugno alle 24:00
  - 1-2/7/25 Ore 8:30 aula Oe - presentazione progetto e iscrizione entro 23 giugno alle 24:00
  - 3-4/9/25 ore 8:30 aula Oe - presentazione progetto e iscrizione entro 26 agosto alle 24:00
  - 2/2/26 ore 8:30 aula Oe - presentazione progetto e iscrizione entro 25 gennaio alle 24:00

Successivamente alla consegna ogni gruppo verrà convocato per discutere il Progetto in un orario preciso.

CONSEGNA

Verifiche finali prima della consegna

- rileggere attentamente le specifiche di progetto
- E’ stato consegnato tutto quello che era richiesto nelle specifiche di progetto?
- I diagrammi sono completi?
- Il codice rispecchia la documentazione prodotta?
- Per eseguire il codice basta fare clone del repository e seguire le istruzioni nel manuale di installazione? (foolproof)

Topics

- Regole generali
- Deliverables
- Valutazione
- Argomento del progetto

DELIVERABLES

6 deliverables **SEPARATI**:

1. un **manuale"** →2-3 pagine. github
2. **Definizione di user stories** → jira
3. un **documento di design** → 4-5 pagine. github
4. il **codice** → github
5. un **documento di system test** → 1-2 pagine. github
6. un **report di unit test** → automatico

DELIVERABLES

un **manuale"** →2-3 pagine

- Una descrizione ad alto livello del Progetto
- le istruzioni su come installare e lanciare il software.
- Indicazioni su ambienti di esecuzione, vincoli su version java, etc.
- Un’indicazione delle principali funzioni riutilizzate da librerie esistenti (escluse quella banali, log4j, java.utils....)
- Indicazione di principali API esterne utilizzate

DELIVERABLES

Definizione di user stories

→ 4-5 (indicativo)

1. **User stories su jira**
2. Descrizione testuale di dettaglio all'interno di ogni user story

As a credit card holder, I want to view my statement balance, so that I can pay the balance due

- Display statement balance upon authentication
- Display Total Balance
- Show "Payment Due Date" and "Minimum Payment Due"
- Display Error message if service not responding/ timeout

DELIVERABLES

documento di design → 4-5 pagine

1. domain model, con una descrizione testuale (se serve)
2. System sequence diagrams
3. design class model
4. Internal sequence diagrams (i più significativi)

DELIVERABLES

codice (su github)

- Codice, files di compilazione, etc.
- opportunamente commentato
- Leggibile e compilabile con un IDE
- All'interno del codice ci devono essere anche le classi di test (junit)

Use design patterns whenever you see appropriate

DELIVERABLES

Documento di system test → 1-2 pagine (indicativo)

- Definizione degli acceptance criteria
  - corrispondono n-1 alle user stories.
  - Vengono testati manualmente dall'interfaccia utente.
- System test report: è un doc che riporta il risultato della validazione di ciascun acceptance criteria (ok/ko, commenti, data)

Example 1: User story and it's acceptance criteria:

As a credit card holder, I want to view my statement (or account) balance, so that I can pay the balance due.

the acceptance criteria for this story could be:

- Display statement balance upon authentication. Say for example $1560
- Display total balance. For example $3560. Here the balance due from the current period is $2560 and past balance due is $2000.
- Show **Minimum payment due**. For example $140
- Show **Payment due date.** For example May 16th of the current month
- Show error message if service not responding or timeout. For example 'Sorry, something went wrong with the service. Please try again.'

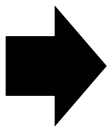
DELIVERABLES

unit test report

- Gli unit test case sono già nel codice, come classi di test. Di solito corrispondono 1-1 alle classi software significative.

- il report è quello generato automaticamente da junit

Topics



- Regole generali
- Deliverables
- Valutazione
- Argomento del progetto

VALUTAZIONE

La valutazione del progetto avverrà tenendo conto dei seguenti punti:

- La realizzazione delle specifiche funzionali;
- La coerenza nell'uso della metodologia
- L'organizzazione e la leggibilità del codice;
- L'adeguatezza della documentazione allegata al progetto
- La corretta definizione dei test
- La presentazione del Progetto

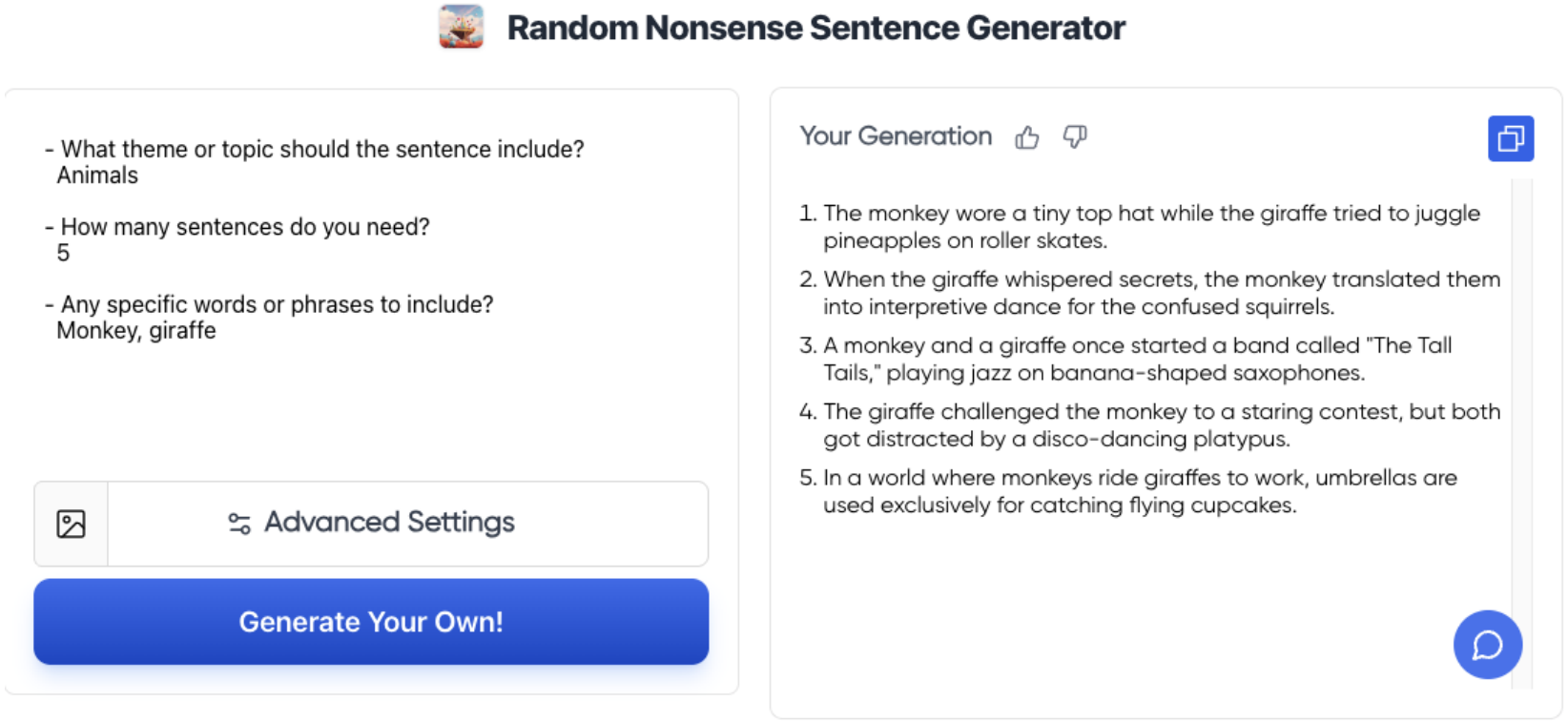
La valutazione individuale dipende dall'efficacia di ciascuno nella presentazione del Progetto e da alcune domande individuali sulla parte teorica.

Topics

- Regole generali
- Deliverables
- Valutazione
- Argomento del progetto

NONSENSE generator

Example:



Basic functionalities

.
Italian/English allowed

- Take a sentence as input
- Analyze the sentence
  - Validate the sentence structure and (on request) provide the syntactic tree.
  - Extract nouns, verbs, adjectives form the sentence.
  - https://language.googleapis.com/v1/documents:analyzeSyntax
- Randomly generate a correct sentence template
  - Es: The [noun] [verb] the [adjective] [noun] in a [adjective] [noun] and [sentence]
  - Es: https://academicguides.waldenu.edu/writing/grammar/sentences
- Select some additional terms from a built-in dictionary of verbs, nouns, adjectives
- Fill-in the template with the terms from the input+additional terms
- Validate the resulting sentence toxicity and show validation results
  - https://language.googleapis.com/v1/documents:moderateText

AI generator (don’t do that)

- Es: https://picoapps.xyz/builder
- https://lovable.dev/
- https://horizons.hostinger.com/
- https://bubble.io/ai-app-generator

But use it to test your ideas…

Do that

- Create files including lists of
  - Nouns
  - Verbs
  - Adjectives
  - SentenceStructures
- Create classes for such objects
  - At init time each class loads the list
  - The constructor may return a random element from its list

Optional

- Select a time for the sentence to be produced
  - present/past/future
- Save the resulting sentences/morphology to a bucket
  - data can be coded as json/xml/txt files
- Besides random generation of a correct sentence template,
  - allow to select from a list of pre-defined templates
- Allow to add terms from the input sentence to the dictionary
  - so that they will be available next time
- Any additional feature you deem relevant…

Sentence analyzer

https://cloud.google.com/natural-language/docs/analyzing-syntax?hl=en

https://cloud.google.com/natural-language/docs/reference/rest/v1/documents/analyzeSyntax (for trying API)

https://cloud.google.com/natural-language/docs/morphology

Risorse

- https://cloud.google.com/storage/docs/reference/libraries#client-libraries-install-java (for java client libraries)
- https://console.cloud.google.com/apis/api/language.googleapis.com/metrics?hl=en (to control API usage)
- https://cloud.google.com/natural-language/pricing (use the free tier)
- Github
- Chatgpt/copilot ok if they fit with your design model
- https://en.wikipedia.org/wiki/ELIZA (for inspiration)

800 ANNI
1222 • 2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA