UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Software Engineering project for the final course exam

A.A. 2023-24

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/1oOMZ8aHM_x7QMunTM2Jyi784pP1G50HssfscdQx94hI/edit?usp=sharing

  (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 come gdoc, pagine wiki all'interno del Progetto, etc.
- Per i diagrammi UML si può usare un tool (plantuml, etc ....)
- Si consiglia di utilizzare IntelliJ 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):
  - 1 appello: 18-19 giugno 2024 - presentazione progetto e iscrizione entro 10 giugno alle 24:00
  - 2 appello: 9-19 luglio 2024 - presentazione progetto e iscrizione entro 1 luglio alle 24:00
  - 3 appello: 5-6 settembre 2024 - presentazione progetto e iscrizione entro 28 agosto alle 24:00
  - 4 appello: 6-7 febbraio 2025 - presentazione progetto e iscrizione entro 29 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? Nel diagramma delle classi tutte le associazioni hanno un nome e cardinalità? Le classi hanno gli attributi necessari?
- Il codice rispecchia la documentazione prodotta?
- Per eseguire il codice basta fare checkout del repository e seguire le istruzioni nel manuale di installazione?

Topics

- Regole generali
- Deliverables
- Valutazione
- Argomento del progetto

DELIVERABLES

6 deliverables **SEPARATI**:

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

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

- domain model, con una descrizione testuale (se serve)
- System sequence diagram
- design class model
- 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 1-1 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

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....)

Topics

- Regole generali
- Deliverables
- Valutazione
- Argomento del progetto

VALUTAZIONE

La valutazione del progetto avverrà tenendo conto dei seguenti punti:

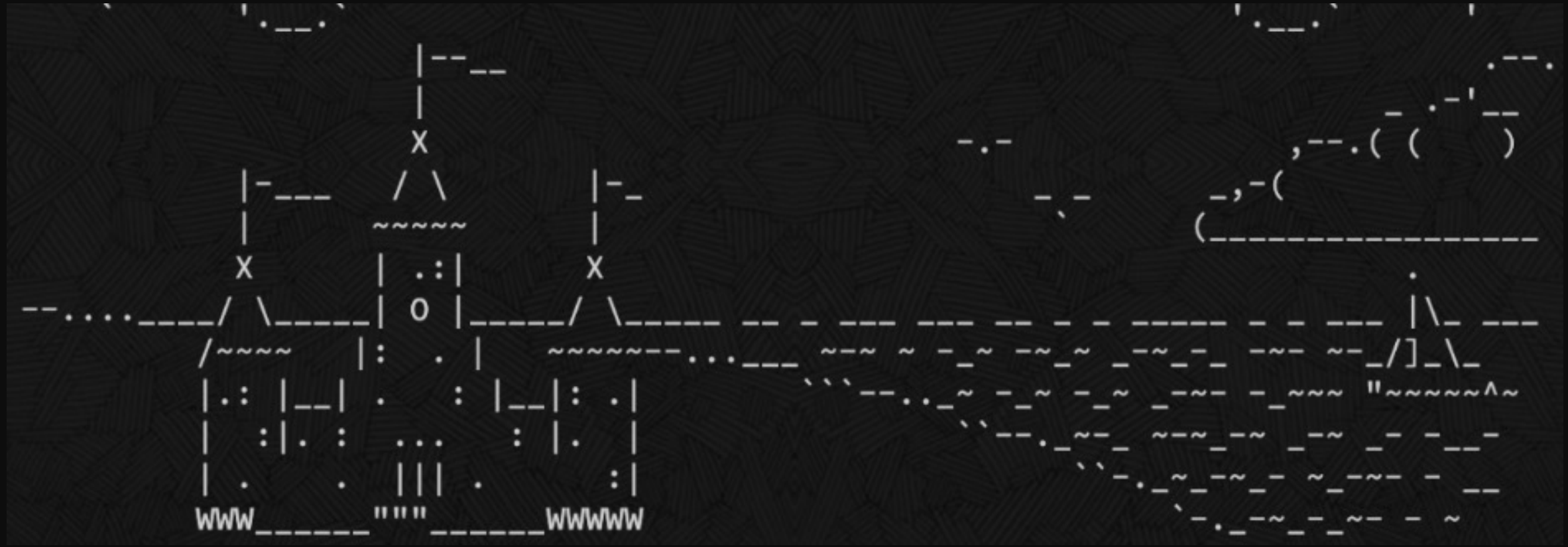
- La realizzazione delle specifiche funzionali;
- L'organizzazione e la leggibilità del codice;
- L'adeguatezza della documentazione allegata al progetto
- La corretta definizione dei test
- La discussione del Progetto

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

PROGETTO TEXT-ADVENTURE



Text adventure

https://en.wikipedia.org/wiki/Text-based_game

Esempi: https://github.com/search?q=java%20text%20adventure&type=repositories

The story

- Define the setting
  - The scenery (a dungeon, a city, a building, etc.)
  - The locations (rooms, streets, caves....)
- Put objects in the locations (people, monsters, axes, etc.)
- Define the goal of your game.
  - Example 1: You are lost in a dungeon. You must find the exit to win.
  - Example 2: You are at unipd. You have to find in which room your Software engineering Exam is being held to win
- Define the rules of your game.
  - Example 1: You find several items in a room. You are only allowed to take one. You can only take one if you did not noreach the maximum weight.
  - Example 2: You meet a dwarf. If you find something to eat that you can give to the dwarf, then the dwarf tells you where to find a magic wand. If you use the magic wand in the big cave, the exit opens....
  - Example 2: You are in the student front office. You must present your student card to get an information.

Basic functionalities

- The game has several “rooms”. Rooms are adjacent (north/south/east/west)
- The player can walk through the rooms, but not all passages are allowed.
- There are items in some rooms. Every room can hold any number of items. Some items can be picked up by the player, others can’t.
- The player can carry some items with them. Every item has a weight. The player can carry items only up to a certain total weight.
- (possibly) the player has a “score” and a “health”
- The player can win. There has to be some situation that is recognised as the end of the game where the player is informed that they have won.
- Implement the commands “north”, “south”, “east”, “west”, ”look (room for items)”, “take <item>” “use <item>” and possibly others you invent.
- Implement a command “back” that takes you back to the last room you’ve been in.
- Implement commands “save”, “exit”, “load”.

Optional functionalities

- Add characters to your game. Characters are people, animals, monsters... – anything that moves. Characters are also in rooms (like the player and items). Unlike items, characters can move around by themselves.
- Extend the parser to recognise three-word commands. You could, for example, have a command: *give bread dwarf* to give some bread (which you are carrying) to the dwarf.
- Add a magical transporter room – every time you enter it you are transported to a random room in your game.
- Others. You can invent additional challenge tasks for yourself.
- Visual rendering (ascii or graphical)

Save/load

- To save/load games use google cloud buckets or aws s3 buckets
- There must be one object listing all avaiable games + one object for each game
- Games are coded as json/xml/txt files

Risorse

- https://cloud.google.com/storage/docs/reference/libraries#client-libraries-install-java (if you use google cloud)
- Github
- Chatgpt

800 ANNI
1222 • 2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA