Esercizio: `DeptRoom`

- Sia definita una classe **`DeptRoom`**, che rappresenta una stanza di un dipartimento universitario. La classe ha:
  - campi: numero, numero di telefono, nome della destinazione, variabile boolean **`isLab`** (dice se è un laboratorio). Il primo è un **`int`** in formato XYY (X=piano, YY=numero progressivo). Il secondo è una stringa, e “parte” da +39 0532 555500.
  - metodi: per accedere e settare i valori, e un opportuno metodo **`toString()`**

Esercizio: `DeptRoom`

- Si consideri ora una classe **`DeptFloor`**, che rappresenta un piano del dipartimento.
- Il piano contiene un numero fissato di stanze **`ROOMS_PER_FLOOR`** (per esempio 25).
  - Ogni piano ha il suo numero, in ordine crescente a partire da 0 (piano terra)
  - La prima stanza (scegliere se è 0 o da 1; è una buona idea documentarlo) ha sempre come destinazione “Sala Riunioni piano N”
  - Scrivere costruttori e metodi

Esercizio: `DeptRoom`

- Da ultimo scrivere una classe che utilizza le struture appena create
  - “Popola” le stanze del dipartimento, partendo dal piano terra. Chiede input per una stanza di quel piano (prima numero, poi destinazione) oppure passa al piano superiore, creato al momento
  - “Interroga” la struttura dati così riempita usando alcuni metodi di cui viene chiesto da input i parametri di ingresso e viene ritornato il valore

Esercizio: `DeptRoom`

- I metodi di “interroga” che vanno implementati sono (se ne possono aggiungere altri, documentandoli):
  - chiedi il numero di un piano e ritorna il numero di laboratori che ci sono a quel piano
  - chiedi il numero di un piano e ritorna un valore boolean che dice se ci sono almeno due stanze con la stessa destinazione (tutelandosi con riempimenti sbadati nell’usare le maiuscole)
  - chiedi una stringa e ritorna tutte le stanze che contengono quella stringa nella destinazione

Progetto: Simulation game

- Gioco che riproduce attività della vita reale
  - In forma semplificata e tramite valori numerici
- Basato su strategia/pianificazione
  - Allocazione di risorse in diverse azioni
  - Nessuna AI da sconfiggere, incognite da scoprire
- Scenari possibili:
  - Militare
  - Manageriale
  - Vita quotidiana
  - Sentimentale

Progetto: Simulation game

- Schema tipico di un “simulation game”
- Fase iniziale: creazione personaggio
  - Scelta di nome ed eventuali attributi numerici oppure rappresentati come lista (inventario)
- Fase principale a turni (turno = giorno, anno...)
  - Le mosse per turno sono limitate da un budget (es.: stanchezza, oppure numero fisso di mosse)
  - Non consideriamo una versione real-time
- La fase continua fino al raggiungimento di un obiettivo, oppure per un limite massimo di turni

Progetto: Simulation game

- Durante ogni turno il giocatore può scegliere le mosse da un insieme possibile.
  - Consigliato introdurre “stanze” per semplificare il menù delle azioni; per ogni stanza sono possibili solo alcune azioni, o spostarsi ad altre stanze
- Esempi di mosse tipiche:
  - allenare il protagonista
  - acquistare oggetti
  - interagire con altri personaggi
- Corrispondono tutte a potenziare alcuni valori a scapito di altri

Progetto: Simulation game

- Quali caratteristiche di Java occorre usare?
- Sicuramente **ereditarietà** / **polimorfismo**.
- Dovrebbero anche essere inseriti, in ordine di importanza: **lettura/scrittura** da file (salvare la partita) **eccezioni**, **thread**; tutti argomenti che verranno discussi nel resto del corso.
- A piacere, si può fare uso di grafica, collezioni... ogni scelta è libera.

Progetto: Simulation game

- Per prima cosa, pensare all'ambientazione e alla meccanica del gioco (tenerla semplice).
- Esempi di trama:
  - Diventare il miglior calciatore della Serie A
  - Far vincere al proprio cane la mostra canina
  - Vincere un reality show
  - Colonizzare un pianeta alieno
  - Conquistare un potenziale partner a una festa

Progetto: Simulation game

- Dopo aver deciso l'obiettivo:
  - Quali sono le caratteristiche del protagonista? (Tipicamente: attributi + riserve di energia; non dev'essere per forza un essere umano)
  - Come si consegue l'obiettivo? (Si prova dopo N turni, o quando è raggiunta una condizione?)
  - Ci sono obiettivi intermedi?
  - Elencare le azioni possibili e magari suddividere i menù in modo logico (es. tramite stanze)
  - Pensare a come usare l'ereditarietà e/o il polimorfismo

Esempio

- Obiettivo: l'unico esame che ti manca per laurearti farà l'ultimo appello tra un mese!
  - Caratteristiche del protagonista, divise tra attributi (intelligenza, fortuna..) e riserve (energia, soldi..)
  - Un attributo speciale determina la conoscenza dell'esame da parte del protagonista
  - 1 turno = 1 giorno; ogni giorno si resetta l'energia a 100; ogni settimana si ricevono 100 euro
  - Al termine del mese di tempo, si prova a dare l'esame e bisogna prendere almeno 18

Altri suggerimenti per estensioni

- La creazione personaggio di solito consente di settare attributi iniziali diversi
  - Si può pensare di avere giocatori diversi per: “classe” (classi diverse pososno fare azioni diverse) e “livello” (migliori capacità di farle)
- Durante il gioco ci possono essere indovinelli
  - Si può pensare che la risposta debba essere nota al giocatore (se in stile con il gioco) oppure che venga disseminata quale indizio durante il gioco

Altri suggerimenti per estensioni

- Si possono inserire oggetti in un inventario del protagonista (a seconda dei casi, diverse implementazioni sono possibili).
  - Oppure, non solo oggetti, ma anche incantesimi, mosse di combattimento, risoluzione di enigmi...
- Alcune parti possono essere interattive
  - Si possono implementare scontri, mini-giochi, o domande con risposta a tempo; possono richiedere di possedere oggetti specifici.