

HOMEWORK A - 2025

Introduzione a Java
Testing

Esercizio 1 (Testing)

- Scrivere le classi di test JUnit per le classi **Stanza**, **Partita**
- Scrivere almeno tre metodi di test per ciascuno dei metodi più importanti di queste classi
- Perseguire la qualità dei test, in particolare la loro *minimalità*. Allo scopo scrivere test-case
 - brevi e coincisi: che utilizzino meno oggetti possibile ed in uno stato iniziale più semplice possibile
 - fattorizzati: senza codice duplicato ma evitando di comprometterne leggibilità ed autocontenimento
 - indipendenti: che evitino di ripetere scenari di test già coperti da altri test-case

Esercizio 2 (Refactoring)

- Introdurre la classe **Labirinto**
 - ha la responsabilità di creare il labirinto, di memorizzare la stanza iniziale (entrata) e quella finale (uscita)
 - aggiungere un riferimento ad un'istanza di **Labirinto** nella classe **Partita** (che ovviamente dovrà essere liberata dalle responsabilità spostate nella nuova classe)
- Introdurre la classe **Giocatore** e la classe **Borsa**
 - **Giocatore** ha la responsabilità di gestire i CFU del giocatore e di memorizzare gli attrezzi in un oggetto istanza della classe **Borsa** >>(vedi codice a seguire)
 - aggiungere un riferimento ad un'istanza di **Giocatore** nella classe **Partita** (che ovviamente dovrà essere liberata dalle responsabilità spostate nella nuova classe)
- Nell'ordine:
 - scrivere classi di test JUnit per **Giocatore**, **Borsa** e **Labirinto**
 - introdurre le classi **Labirinto** e **Giocatore** e **Borsa** nel codice

Esercizio 3 (Refactoring, Nuovi Comandi)

- Modificare il gioco affinché il giocatore possa "prendere" e "posare" degli attrezzi
- Per implementare questa modifica è necessario fare le seguenti operazioni:
 - completare **Stanza** ed il suo metodo **removeAttrezzo()**
 - utilizzare il codice di **Borsa** riportato di seguito completando il metodo **removeAttrezzo()**
 - modificare **DiaDia** implementando il codice per l'esecuzione dei comandi **prendi** e **posa**
 - gli attrezzi presi vengono rimossi dalla stanza e aggiunti alla borsa
 - gli attrezzi posati vengono rimossi dalla borsa e aggiunti alla stanza
 - la sintassi per inserire questi comandi è la seguente:
 - **prendi** <nomeAttrezzo>
 - **posa** <nomeAttrezzo>
 - modificare la logica del comando «aiuto» che deve tener conto del nuovo comando disponibile

Classe Borsa: Codice (1)

```
public class Borsa {
    public final static int DEFAULT_PESO_MAX_BORSA = 10;
    private Attrezzo[] attrezzi;
    private int numeroAttrezzi;
    private int pesoMax;

    public Borsa() {
        this(DEFAULT_PESO_MAX_BORSA);
    }

    public Borsa(int pesoMax) {
        this.pesoMax = pesoMax;
        this.attrezzi = new Attrezzo[10]; // speriamo bastino...
        this.numeroAttrezzi = 0;
    }

    public boolean addAttrezzo(Attrezzo attrezzo) {
        if (this.getPeso() + attrezzo.getPeso() > this.getPesoMax())
            return false;
        if (this.numeroAttrezzi==10)
            return false;
        this.attrezzi[this.numeroAttrezzi] = attrezzo;
        this.numeroAttrezzi++;
        return true;
    }

    public int getPesoMax() {
        return pesoMax;
    }

    public Attrezzo getAttrezzo(String nomeAttrezzo) {
        Attrezzo a = null;
        for (int i= 0; i<this.numeroAttrezzi; i++)
            if (this.attrezzi[i].getNome().equals(nomeAttrezzo))
                a = attrezzi[i];
        return a;
    }
}
(continua)
```

Classe Borsa: Codice (2)

```
public int getPeso() {
    int peso = 0;
    for (int i= 0; i<this.numeroAttrezzi; i++)
        peso += this.attrezzi[i].getPeso();
    return peso;
}

public boolean isEmpty() {
    return this.numeroAttrezzi == 0;
}

public boolean hasAttrezzo(String nomeAttrezzo) {
    return this.getAttrezzo(nomeAttrezzo)!=null;
}

public Attrezzo removeAttrezzo(String nomeAttrezzo) {
    Attrezzo a = null;
    // ---> TODO (implementare questo metodo) <---
    return a;
}

public String toString() {
    StringBuilder s = new StringBuilder();
    if (!this.isEmpty()) {
        s.append("Contenuto borsa ("+this.getPeso()+"kg/"+this.getPesoMax()+"kg): ");
        for (int i= 0; i<this.numeroAttrezzi; i++)
            s.append(attrezzi[i].toString()+" ");
    }
    else
        s.append("Borsa vuota");
    return s.toString();
}
}
```

Esercizio 4 (Package)

- Il progetto sta crescendo: organizziamo meglio le classi in package
 - mettere **Labirinto** e **Stanza** nel package `it.uniroma3.diadia.ambienti`
 - mettere **Attrezzo** nel package `it.uniroma3.diadia.attrezzi`
 - mettere **Giocatore** e **Borsa** nel package `it.uniroma3.diadia.giocatore`
 - mettere **Comando**, **DiaDia** e **Partita** nel package `it.uniroma3.diadia`

Esercizio 5

(Disaccoppiamento I/O)

- La gestione dell'Input e dell'Output è attualmente “disseminata” in molti punti del codice
 - uso diretto di **System.out** (per la stampa su video)
 - uso diretto di **System.in** (per la lettura da tastiera)
- Questa situazione non è ottimale
 - difficile trovare e cambiare i messaggi stampati
 - dentro un test-case risulta poco agevole
 - sia controllare le stampe con delle asserzioni
 - sia iniettare dei comandi scelti nel test stesso
 - sarà difficile cambiare la modalità di interazione con l'utente (ad es. per passare ad una modalità alternativa basata sull'utilizzo di GUI)

Esercizio 5

➤ Riorganizziamo completamente la gestione dell'I/O, disaccoppiando il gioco dall'uso diretto e pervasivo di **System.out/System.in**, operando come segue:

- Introduciamo la classe **it.uniroma3.diadia.IOConsole** (fornita nella prossima slide) che centralizza l'accesso a **System.out/System.in**
- Creiamo una sola istanza di questa classe nell'unico metodo **main()** dell'intero gioco di ruolo **DiaDia.main()**
- Rifattorizziamo tutto il codice per far arrivare tale unica istanza ove serve
 - Può essere necessario aggiungere costruttori e/o modificare quelli già esistenti in alcune classi
- Eliminiamo tutte le stampe e le letture dirette dal resto del codice rimpiazzandole con un appropriato uso dei metodi della classe **IOConsole**

Esercizio 5: IOConsole - ATTENZIONE

- NON può essere modificato il codice di **IOConsole**
- NON può essere modificato il suo package
- Crearne UNA SOLA ISTANZA in tutto il codice nel metodo **DiaDia.main()**
- NON è più possibile usare **System.in** e **System.out** altrove

```
package it.uniroma3.diadia;  
import java.util.Scanner;
```

```
public class IOConsole {
```

```
    public void mostraMessaggio(String msg) {  
        System.out.println(msg);  
    }
```

```
    public String leggiRiga() {  
        Scanner scannerDiLinee = new Scanner(System.in);  
        String riga = scannerDiLinee.nextLine();  
        scannerDiLinee.close();  
        return riga;
```

Per il momento omettere
questa riga, ci torneremo

```
    }
```

```
}
```

TERMINI E MODALITA' DI CONSEGNA

- La soluzione deve essere inviata al docente entro le 21:00 di domenica 6 aprile 2025 come segue:
 - Svolgere in gruppi di esattamente 2 persone
 - L'oggetto (subject) *DEVE* iniziare con la stringa **[2025-HOMEWORKA]** seguita dalle matricole
Ad es.: **[2025-HOMEWORKA] 612345 654321**
- Modalità di consegna tramite GitHub descritta di seguito (>>)
 - Creare un repository Github
 - Caricare il sorgente sviluppato
 - Creare una release di nome **versione.A**
 - Inviare una email contenente un link alla release del codice

MODALITA' DI CONSEGNA

Per consegnare usare
questo indirizzo di email!
Gestita automaticamente.

- Per la consegna utilizzare
poo.roma3@gmail.com
- Non inviare mail di richiesta di conferma di avvenuta ricezione, non saranno gestite
 - Eventuali problematiche saranno gestite successivamente, quando e se *veramente* serve
- Consegne multiple da evitare ed inutili
 - ✓ Inutile: fa fede la data della release non dell'email
 - ✓ Ma solo le email verranno gestite automaticamente per raccogliere le matricole di chi ha consegnato
- ATTENZIONE:
 - NON allegare direttamente archivi .jar, .zip, .tar.gz
 - ✓ Per motivi di sicurezza l'email non verrebbe consegnata
 - Aggiungere nel corpo del messaggio:
 - Il link al repository github (>>)
 - Descrizione degli eventuali malfunzionamenti noti, ma non risolti

TERMINI E MODALITA' DI CONSEGNA

- ATTENZIONE:
 - Senza l'invio di questo homework, *non* sarà possibile continuare il percorso HQ di questo anno accademico

Consegna con GitHub (1)

- GitHub è un servizio di hosting di progetti versionati con *git*
 - Darà ospitalità ai sorgenti dei nostri progetti
- Principali obiettivi del servizio
 - permettere la cooperazione di più sviluppatori condividendo i sorgenti dello stesso progetto
 - conservare la storia del progetto e di tutte le versioni dei sorgenti mai prodotte

Qualche riferimento (fate anche qualche ricerca...):

- <https://github.com/git-guides>

(ma si trovano anche interi libri...)

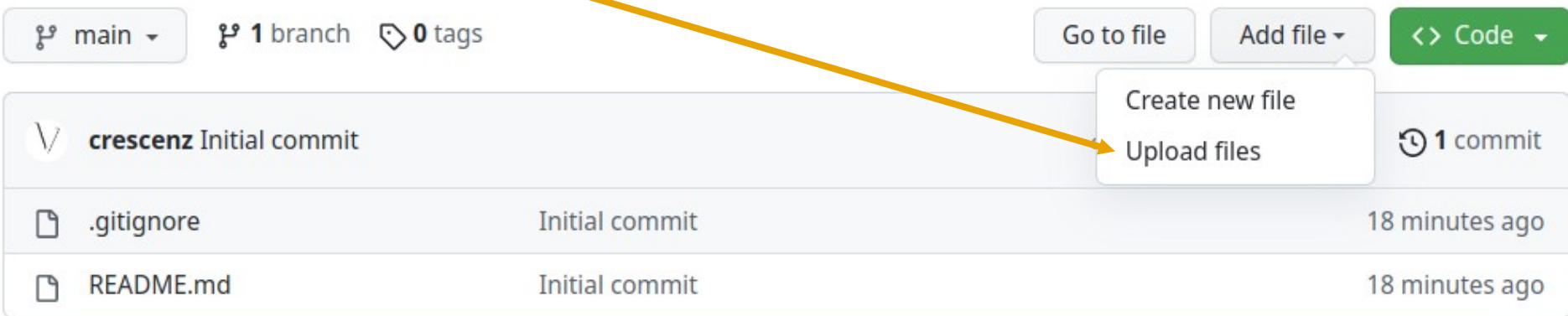
- L'obiettivo è cominciare ad usare *git*
 - per ora solo uno strumento che faciliti le consegne
 - nel futuro, strumento *abilitante* il lavoro cooperativo, in team
- L'utilizzo efficace di *git* **NON** è tra gli obiettivi formativi del corso
 - ✓ Il suo utilizzo è fortemente *incentivato*...
 - ...ma *richiesto* solo come strumento per la consegna

Consegna con GitHub (2)

- Caricare il progetto su GitHub. Faremo una prima *release* (>>)
 - Registrarsi su *github.com*
 - Creare un nuovo repository (pubblico) per ospitare il codice del vostro primo progetto
 - cliccando su «create repository»
 - Il nome della repository dovrà essere
 - 2025-HOMEWORK-«MATRICOLA1»-«MATRICOLA2»
 - ad es.: 2025-HOMEWORK-612345-654321
 - *Git Repository*
 - il luogo dove i file del sorgente del progetto sono “versionati”
 - ovvero viene conservata traccia di tutta la loro «storia» dal momento in cui sono caricati all'interno del repository ad oggi
 - accessibile da altri utenti e modificabile dagli altri *collaboratori*

Consegna con GitHub (3)

- Una volta creato il repository, caricare il codice sorgente del progetto seguendo le istruzioni che leggete su schermo, oppure, ancora più semplicemente:
 - Nella pagina della repository appena creata, utilizzare: upload files (e quindi >>)



main 1 branch 0 tags

Go to file Add file <> Code

crecenz Initial commit 1 commit

.gitignore	Initial commit	18 minutes ago
README.md	Initial commit	18 minutes ago

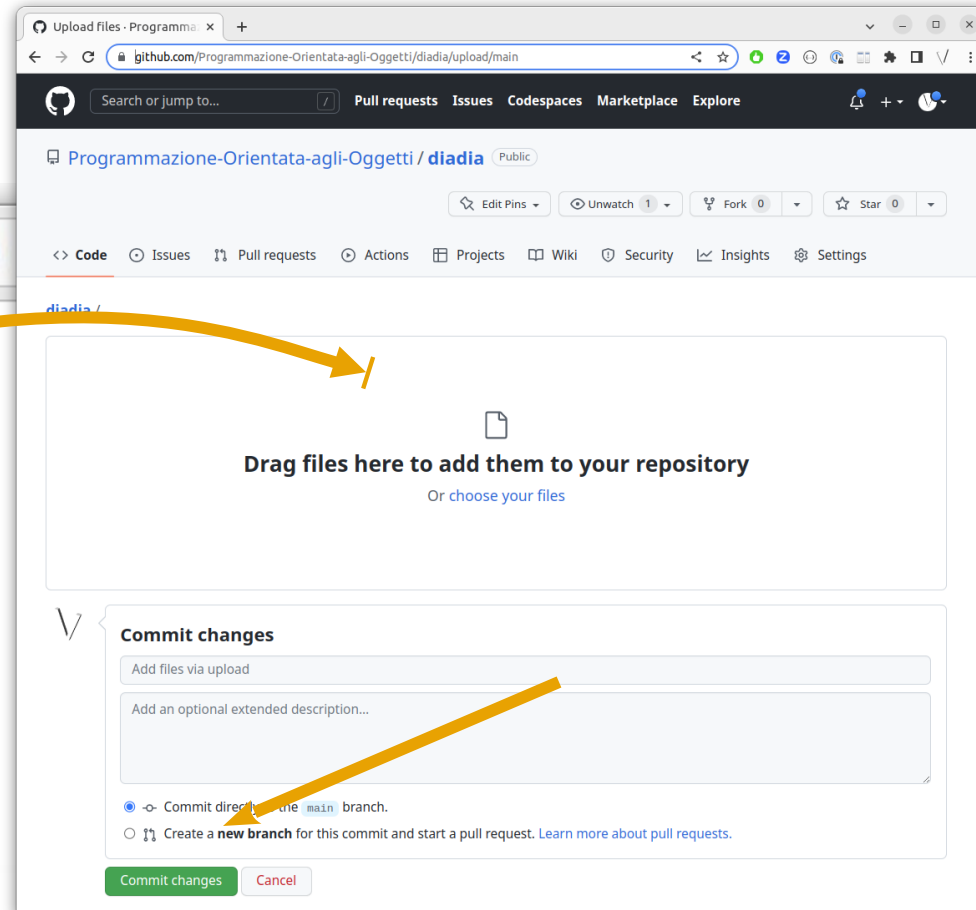
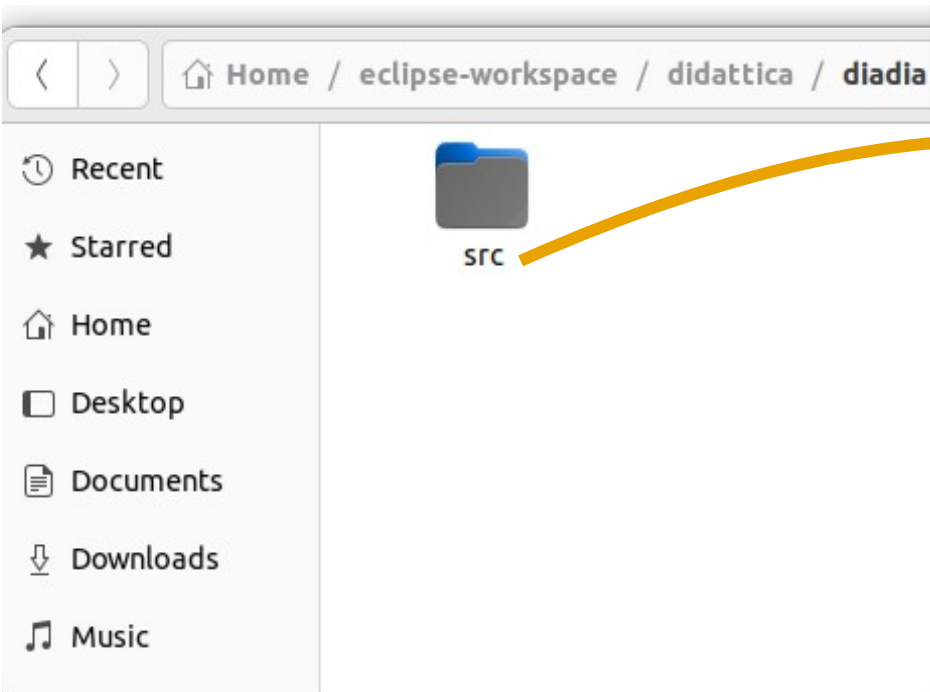
README.md

2025 - HOMEWORK - 612345 - 654321

Lo Studio di Caso a Supporto del Corso di Programmazione Orientata agli Oggetti

Consegna con GitHub (4)

- Navigare fino alla directory del vostro progetto
 - ad es. `/eclipse-workspace/didattica/diadia/`
 - Trascinare *tutta* la cartella `src` dentro il git repository appena creato
 - Successivamente ripetere l'operazione con la cartella `test`
- Quindi salviamo le modifiche
 - tasto verde *Commit changes*



Consegna con GitHub (5)

- Per finire, creare una *release*: “impacchettamento” del progetto per renderlo disponibile ad altri
 - Selezionare <<create new release>> a dx nella pagina

The screenshot shows a GitHub repository named 'crescenz' with the description 'Versione iniziale (base)'. The repository has 3 commits, the latest being 1 minute ago. The file list includes 'src' (7 minutes ago), '.gitignore' (1 hour ago), and 'README.md' (1 hour ago). The README content is visible, showing the repository name 'diadia' and the course name 'Lo Studio di Caso a Supporto del Corso di Programmazione Orientata agli Oggetti'. On the right sidebar, under the 'Releases' section, it states 'No releases published' and provides a blue link 'Create a new release'. A yellow arrow points from the text 'Selezionare <<create new release>> a dx nella pagina' to this link.

- Scegliere il cosiddetto *tag* della release
 - Usare **versione.A**
 - Come nome della release inserire
 - 2025-HOMEWORKA <<matricola1>> <<matricola2>>
 - Ad es. 2025-HOMEWORKA 612345 654321
- Nella descrizione scrivere eventuali malfunzionamenti noti ma non risolti

TERMINI E MODALITA' DI CONSEGNA

- Esempio di come appare la release della versione base pubblicata
- ✓ Copiare l'URL dell'archivio (NON l'archivio) e mandarlo via mail come già spiegato

The image shows two side-by-side screenshots of a GitHub release page for a repository named 'crescenz'. The release is titled 'Versione base' and is marked as 'Pre-release'. It was released 'this now' with version 'v0.0.0' and commit '030636e'. The description states 'La versione iniziale (base) del codice'. Under the 'Assets' section, there are two items: 'Source code (zip)' and 'Source code (tar.gz)'. A yellow arrow points from the 'Source code (tar.gz)' link in the left screenshot to the 'Source code' link in the right screenshot. In the right screenshot, a context menu is open over the 'Source code' link, showing options like 'Open link in new tab', 'Open link in new window', 'Open link in incognito window', 'Save link as...', and 'Copy link address'. The 'Copy link address' option is highlighted. The footer of the page shows the GitHub logo, '© 2023 GitHub, Inc.', and a 'Status' link.