PROGETTO MULTIDISCIPLINARE TPS - INFORMATICA

In base al progetto assegnato, lo studente, seguendo gli step previsti dal ciclo di vita del software, dovrà progettare, realizzare, documentare e testare un software applicativo che consenta di gestire una struttura dati (array/array list) opportuna.

- **Sia il codice sia la documentazione prodotta (ad esempio i diagrammi UML, i piani di test e tutto quanto ritenuto utile per la realizzazione del progetto) dovranno essere caricato su Github sin dal primo giorno e aggiornato almeno ad ogni sessione di lavoro (ogni volta che ci sarà una lezione di Informatica o di TPS). Questo rientra nella valutazione!**

- Il software applicativo, all'avvio deve presentare un MENU che consenta di svolgere le operazioni richieste dal progetto.

- La tabella dei requisiti e il diagramma dei casi d'uso vanno caricati il primo giorno di lavoro in classe.

- In caso ci fossero dei dubbi su cosa deve fare il software, lo studente può chiedere al docente per dirimere con esso tali dubbi. Il docente rappresenta "l'esperto del dominio" che ha commissionato il software. Il software deve soddisfare almeno i requisiti minimi stabiliti con il docente che sono: aggiunta, ricerca di un elemento, aggiornamento, eliminazione di un elemento, un ordinamento degli elementi, esportazione su file CSV, importazione da file CSV, serializzazione e deserializzazione. Lo studente è sempre libero di aggiungere ulteriori funzionalità. La classe principale deve contenere almeno un attributo di classe LocalDate, LocalTime o LocalDateTime.

- Si consiglia di realizzare il diagramma delle classi prima di iniziare la stesura del codice e di AGGIORNARLO AL TERMINE DEL PROGETTO Si ricorda infatti che le classi subiranno necessariamente delle modifiche durante la fase di implementazione quindi il diagramma delle classi finale dovrà corrispondere con le classi effettivamente implementate.

- Complessivamente il tempo a disposizione per la realizzazione del progetto è di due settimane.

- Il progetto deve contenere almeno due classi principali, la classe principale 2 ha come attributo un array o un'array list della classe principale 1. Esempi:

  "Campionato" contiene un arraylist di "Partita"

  "Concessionario" contiene un arrayList di "Automobile"

  "GestorePalestraComunale" contiene un arrayList di "Prenotazione"

- La classe principale (negli esempi le classi "Partita", "Automobile", "Prenotazione") deve contenere almeno un attributo di classe LocalDate, LocalTime o LocalDateTime

FASI DI LAVORO

1. **Analisi**:
    a. Produzione tabella di analisi dei requisiti e Diagramma dei casi d'uso.
    b. Creazione del repository su GitHub.

2. **Progettazione**:
    a. Prima stesura del diagramma delle classi

3. **Implementazione**:
    a. Scrittura del codice
    b. Debug del codice
    (si consiglia di procedere una classe alla volta)

4. **Testing**:
    a. Preparazione del piano di test delle due classi principali.
    b. Realizzazione ed esecuzione del test case con compilazione degli esiti
    c. Preparazione del piano di test di integrazione.
    d. Esecuzione del piano di test di integrazione e compilazione degli esiti.

5. **Documentazione:**
    a. Riscrittura del diagramma delle classi, dell'elenco dei requisiti e del diagramma dei casi d'uso in conformità alle classi realmente realizzate.
    b. Produzione della documentazione con Javadoc delle classi utilizzate.

6. **Nota: il docente si riserva di chiedere chiarimenti, delucidazioni, spiegazioni nella fase di valutazione del progetto. Lo studente è pertanto tenuto a dimostrare di conoscere gli strumenti informatici utilizzati. La griglia di valutazione è riportata in seguito per ciascuna delle due materie.**

SUGGERIMENTI

- Il lavoro può essere svolto in maniera più semplice (generalmente due sole classi) oppure più complesso. Al crescere della complessità aumenterà il punteggio ottenuto. Si consiglia di svolgere inizialmente un progetto semplice che funzioni e poi eventualmente realizzare un refactoring che ne migliori l'aderenza ai principi della OOP (ad esempio aumentando il numero di classi). Per esempio: se si volesse realizzare un software che gestisce le partite di calcio, inizialmente le due squadre (squadra 1 e squadra 2) possono essere due attributi della classe Partita, successivamente si potrebbe prevedere l'aggiunta di una classe Squadra (Nome, città, colori, trofei vinti, anno di fondazione, ecc..)

GRIGLIA DI VALUTAZIONE SOMMATIVA PER LA MATERIA INFORMATICA					
Livelli		**Ottimo**	**Buono**	**Sufficiente**	**Insufficiente**
		Lavoro svolto in modo completo, preciso e in autonomia, adottando anche soluzioni personali.	*Lavoro svolto in modo completo senza adozione di soluzioni personali*	*Lavoro svolto con errori non gravi o in modo non completo*	*Lavoro svolto con errori gravi, con superficialità o non svolto*
Dimensioni e criteri	**Descrittori**	**Punteggio**	**Punteggio**	**Punteggio**	**Punteggio**
Funzionalità di base	Inserimento	**1**	**0,5**		**0**
	Eliminazione	**1**	**0,5**		**0**
	Ricerca	**1**	**0,5**		**0**
	Modifica	**1**	**0,5**		**0**
	Ordinamento	**2**	**1**	**0,5**	**0**
	Utilizzo degli arrayList (soluzione personale)	**2**		**0**	
Utilizzo del meccanismo delle eccezioni	(creazione di eccezioni personalizzate, utilizzo corretto delle eccezioni unchecked, utilizzo corretto delle eccezioni cheched, nessun utilizzo delle eccezioni)	**2**	**1**	**0,5**	**0**
Esportazione dei dati su file CSV		**1**		**0**	
Importazione dei dati da file CSV		**1**		**0**	
Serializzazione e Deserializzazione	Serializzazione	**1**	**0,5**		**0**
	Deserializzazione	**2**	**0,5**		**0**
Bug e malfunzionamenti		**2**	**2**	**1**	**0**
Punteggi aggiuntivi per complessità del progetto	Ad esempio, presenza di più di un attore, aggiunta di funzionalità particolarmente complesse. Aggiunta di ulteriori classi significative. (soluzione personale)	**Max 3 punti**			
Correttezza ed impegno dimostrate nell'attività in laboratorio e a casa	(intense e costanti, buone, altalenanti, nulle)	**2**	**0,50**	**0,25**	**0**
Tempi:	Rispetto dei tempi di consegna indicati per le attività	**3**	**0,50**	**0**	**0**
PUNTEGGIO TOTALE					…./25
ELABORATO					…/10
PUNTEGGIO PROVA ORALE:	Verrà chiesto di spiegare il funzionamento delle classi realizzate e verrà chiesto di implementare nuove funzionalità.			Valutazione secondo tabella PTOF	…/10

GRIGLIA DI VALUTAZIONE SOMMATIVA PER LA MATERIA TPS

Livelli		**Ottimo**	**Buono**	**Sufficiente**	**Insufficiente**
		Lavoro svolto in modo completo, preciso e in autonomia, adottando anche soluzioni personali.	*Lavoro svolto in modo completo senza adozione di soluzioni personali*	*Lavoro svolto con errori non gravi o in modo non completo*	*Lavoro svolto con errori gravi, con superficialità o non svolto*
Dimensioni e criteri	**Descrittori**	**Punteggio**	**Punteggio**	**Punteggio**	**Punteggio**
Analisi dei requisiti e diagramma dei casi d'uso	(Corretti e prodotti autonomamente, corretti non sempre in maniera autonoma, con errori, completamente sbagliati)	**2**	**1**	**0,5**	**0**
Utilizzo costante di GitHub per il versioning del software	Verrà valutata la documentazione caricata su GitHub al termina di ogni sessione di lavoro. Per ogni sessione verranno assegnati o non assegnati 0,5 punti)	Sessione 1 / Sessione 2 / Sessione 3 / Sessione 4 / Sessione 5 / Sessione 6			
Documentazione del software	Diagramma delle classi	**2**	**1**	**0,5**	**0**
	Dcumentazione html prodotta con Javadoc	**2**	**1**	**0,5**	**0**
Testing	Valutazione del piano di test unitario delle classi principali (le eccezioni sono state testate?, Il metodo equals è stato ridefinito?...)	**2**	**1**		**0**
	Valutazione del test case JUnit delle classi principali (i testi previsti sono stati implementati correttamente?)	**2**	**1**		**0**
	Valutazione del piano di test di integrazione e della sua esecuzione	**2**	**1**		**0**
Tempi:	Rispetto dei tempi indicati per le consegne delle attività	**2**	**0,50**	**0,25**	**0**
PUNTEGGIO FINALE					**…./17** **…/10**