**Consegna 4**

**1)Leggete l’articolo *”Programming Patterns and Design Pattern in Introductory Computer Science Course”* e scrivete un breve resoconto sull’approccio proposto: siete d’accordo con l’approccio proposto? Lo adottereste? Immaginate di proporlo in una classe in cui insegnate: quali sono le potenziali difficoltà?**

1) L'articolo *"Programming Patterns and Design Patterns in the Introductory Computer Science Course"* di Viera K. Proulx esplora un approccio pedagogico per l'insegnamento di informatica introduttiva basata sulla programmazione orientata agli oggetti, utilizzando modelli di programmazione e design patterns. Di seguito è riassunto l'approccio proposto nell'articolo e alcune considerazioni sulla sua adozione in un corso.

**Approccio Proposto:** L'articolo propone un approccio basato su modelli di programmazione e design patterns per insegnare informatica introduttiva. L'obiettivo principale è concentrarsi sullo sviluppo delle capacità di ragionamento e progettazione degli studenti prima che la complessità dei dettagli del linguaggio di programmazione li confonda. L'autore ha sviluppato una serie di modelli che coprono argomenti tipicamente affrontati in un corso introduttivo di informatica e li presenta come tutorial. Ogni modello è strutturato in modo da guidare gli studenti attraverso le fasi di progettazione e implementazione, dal contesto generale alla realizzazione pratica. Il corso è organizzato con conferenze settimanali, discussioni di gruppo e laboratori.

**Considerazioni sull'Adozione:** L'approccio proposto sembra avere benefici significativi per gli studenti di programmazione. Tuttavia, ci sono alcune considerazioni da tenere in considerazione prima di adottarlo in un corso:

**Livello di Conoscenza degli Studenti:** L'articolo sembra adatto a studenti che sono completamente nuovi alla programmazione o hanno conoscenze molto limitate. Per studenti con esperienza precedente in programmazione, alcuni contenuti potrebbero risultare troppo elementari.

**Necessità di Adattamento:** È possibile che sia necessario adattare il contenuto dei tutorial e dei problemi di pratica in base alle specifiche esigenze della classe e del linguaggio di programmazione utilizzato.

**Tempo Richiesto:** L'insegnamento basato su modelli e tutorial può richiedere più tempo rispetto a un approccio tradizionale, quindi è importante valutare se è possibile integrare efficacemente questa metodologia nel corso.

**Valutazione:** La valutazione degli studenti dovrebbe essere progettata in modo da riflettere il loro apprendimento basato su modelli e design patterns.

**Sviluppo delle Capacità di Problema:** L'approccio sembra concentrarsi sullo sviluppo delle capacità di progettazione, ma è importante assicurarsi che gli studenti sviluppino anche le abilità di risoluzione di problemi pratici.

**Disponibilità di Risorse:** Sarà importante garantire che gli studenti abbiano accesso alle risorse necessarie, come i tutorial e i materiali di apprendimento.

In generale, l'approccio proposto sembra interessante e può essere efficace per insegnare le basi della programmazione e del design orientato agli oggetti. Tuttavia, è importante valutare attentamente le esigenze specifiche della classe e adattare l'approccio di conseguenza.

**2)riportate il testo del problema analizzato per l'attività 2) Progettazione soluzione/algoritmo e la soluzione che avete progettato: suddivisione in sotto problemi, pattern algoritmici, elementari e ruoli delle variabili.**

2)Il problema scelto è il seguente

**Swap tra due valori**

Realizzare un programma che acquisisca da tastiera (standard input) due valori interi e li memorizzi nelle variabili varA e varB. In seguito, scambiare il valore di tali variabili e stamparne a schermo il nuovo contenuto. Attenzione: gestire il caso in cui non si riceve un valore intero, il programma non deve terminare.

**Soluzione proposta:**

import java.util.Scanner;

public class SwapValori {

Scanner scanner = new Scanner(System.in);

int varA = 0;

int varB = 0;

boolean inputValido = false;

while (!inputValido) {

System.out.print("Inserisci il valore di varA (deve essere un intero): ");

if (scanner.hasNextInt()) {

varA = scanner.nextInt();

inputValido = true;

} else {

System.out.println("Input non valido. Inserisci un valore intero valido.");

scanner.next(); // Pulisce l'input errato

}

}

inputValido = false;

while (!inputValido) {

System.out.print("Inserisci il valore di varB (deve essere un intero): ");

if (scanner.hasNextInt()) {

varB = scanner.nextInt();

inputValido = true;

} else {

System.out.println("Input non valido. Inserisci un valore intero valido.");

scanner.next(); // Pulisce l'input errato

}

}

System.out.println("Valore di varA prima dello scambio: " + varA);

System.out.println("Valore di varB prima dello scambio: " + varB);

int temp = varA;

varA = varB;

varB = temp;

System.out.println("Valore di varA dopo lo scambio: " + varA);

System.out.println("Valore di varB dopo lo scambio: " + varB);

scanner.close();

}

**Suddivisione in sotto problemi**

Abbiamo individuato diversi sotto problemi principali: il primo è la definizione e l’utilizzo dello scanner per prendere l’input da tastiera (potrebbe essere a sua volta scomposto in due sotto problemi); il secondo è il controllo dell’input; il terzo è lo swap vero e proprio delle variabili.

**Ruoli delle variabili**

**Valore più recente:** Le variabili varA e varB conservano i valori più recenti immessi dall'utente.

**Flag unidirezionale:** La variabile inputValido funge da flag unidirezionale, passando da false a true quando l'utente fornisce un input valido.

**Temporanea:** Le variabili temp e scanner possono essere considerate temporanee. temp è utilizzata temporaneamente per effettuare lo scambio tra varA e varB, mentre scanner è utilizzata temporaneamente per acquisire input dall'utente.

**Pattern elementari**

**Lettura in input finchè non si riceve il valore desiderato:** è quello che chiede l’esercizio, continuare a chiedere finchè non si riceve un intero.

**Pattern algoritmici**

**1)Pattern name:** read from input

**Pattern start point:** every exercise that needs values from input

**Pattern objective:** return values taken from input

define scanner

if(value is valid) var=scanner

**2)Pattern name:** variables exchange

**Pattern start point:** 2 variables

**Pattern objective:** returns 2 variables whose values have been changed

temp=var1

var1=var2

var2=temp

**3)Descrivete anche eventuali critiche (es. sulla formulazione del problema), problemi e considerazioni emerse durante l'attività**

Un punto critico è il fatto di dover trasformare un problema troppo generico in codice, nel senso che molto spesso ci troviamo di fronte algoritmi resi obsoleti, come ad esempio la somma in colonna, utile forse solamente per un allenamento logico/matematico, ma a lato pratico ci sono migliaia di strumenti che riescono ad arrivare allo stesso risultato in un tempo enormemente minore.

Inoltre svolgendo l’attività ci si è resti conto che la formulazione del problema è molto importante non solo per la comprensione da parte degli studenti ma anche per l’analisi da parte del docente. Infatti si può dire che parti come la suddivisione in sotto problemi sono state problematiche da sviluppare proprio a causa della formulazione del problema.