1) GESTIONE DI UN MUSEO.

Scrivete le classi necessarie per la gestione delle opere d'arte dei musei. Un oggetto della classe **Museo** conterrà nome, indirizzo e l'elenco delle opere d'arte presenti nel museo.

Scrivete una gerarchia per rappresentare i seguenti oggetti: **OperaDArte, Scultura, Dipinto**. Tutti gli oggetti devono avere un titolo, un autore (String) e la dataDiCreazione (per semplicità un oggetto della classe String) inizializzati dal costruttore. Ogni **Scultura** ha in aggiunta il materiale e l'altezza mentre ogni **Dipinto** ha la tecnica (olio, tempera, litografia, serigrafia, china) e le dimensioni. Tutti i membri devono essere incapsulati.

La classe **Museo** deve avere un metodo per stampare la lista delle opere contenute e metodi per l'inserimento e la cancellazione di opere d'arte.

Utilizzate le vostre classi in un programma che crea un oggetto **Museo** e mostra un menù interattivo testuale in cui si dà all'utente la possibilità di aggiungere nuovi elementi da tastiera, stampare i dettagli di tutti gli oggetti presenti nel museo, eliminare un elemento individuato in base al titolo, stampare i dettagli di un'opera individuata attraverso il titolo.

2) SIMULAZIONE TRAFFICO MARITTIMO

Scrivete un programma per la simulazione del traffico marittimo in prossimità di un porto.

Si supponga che ogni nave sia caratterizzata da un codice univoco (String) e abbia una posizione in coordinate cartesiane (double), una velocità (double), una direzione (int fra 0 e 359), lunghezza e larghezza (entrambi double) e numero di passeggeri a bordo (int).
Le navi devono avere un metodo avanza(), che aggiorna le coordinate in base alla velocità e alla direzione (vedi suggerimenti).

Scrivete una classe **Nave** con tutti i dati, metodi e costruttori che ritenete opportuni.

Scrivete una classe **Porto** che, attraverso un suo metodo, permette di invocare il metodo avanza() di tutti i natanti presenti nello stretto.
La classe **Porto** deve permettere l'inserimento e l'eliminazione dei vari natanti.
Prevedete nella classe **Porto** anche un metodo per la verifica e prevenzione di collisioni (per semplicità il messaggio di pericolo verrà attivato quando la distanza fra due natanti è inferiore a una certa soglia che fisserete voi arbitrariamente).

Utilizzate le classi di cui sopra in un programma che crea un oggetto **Porto**, gestisce l'interazione con l'utente mediante un menu per inserimento o eliminazione di navi e, quando richiesto, invoca il metodo che fa avanzare tutti i natanti, stampa lo stato di tutte le navi e verifica la presenza di potenziali collisioni.

Suggerimenti:
La nuova posizione della nave si calcola con le formule:

```
x+=velocita*cos(direzione*3.14/180);
y+=velocita*sin(direzione*3.14/180);
```

La distanza tra due punti P0(x0, y0) e P1(x1, y1) si può calcolare nel seguente modo:

```
double dx = x1 – x0;
double dy = y1 – y0;
double distanza sqrt(dx*dx + dy*dy);
```

3) ELEZIONE RAPPRESENTANTI STUDENTI

Scrivere le classi necessarie per la gestione di un seggio elettorale per le elezioni dei rappresentanti degli studenti. Nel seggio possono votare gli studenti il cui elenco è noto. Ciascuno studente esprime la preferenza di un candidato.

Potreste pensare di strutturare il programma con una classe **Seggio**, di cui istanzierete un'oggetto nel *main()*, che al suo interno contiene un elenco di oggetti della classe **Studente** e un elenco di oggetti della classe **Candidato**.
Gli studenti sono caratterizzati da matricola, codice di controllo (entrambi *long*), nome e cognome. I candidati, che sono anche studenti, hanno anche il nome della lista a cui si appoggiano (una *String*).

Scrivete quindi un programma che all'avvio crea un'oggetto seggio e carica i dati di studenti e candidati (prevedete semplicemente nel costruttore di default la creazione di tre candidati e una decina di studenti).
Dopo la creazione dell'oggetto il programma entra in un ciclo infinito in cui mostra una schermata che chiede la matricola e il codice di controllo (che è stato dato all'elettore al posto della tradizionale scheda elettorale cartacea), verifica che matricola e codice di controllo siano coerenti, stampa l'elenco dei candidati come elenco numerato, permette all'elettore di esprimere la sua preferenza mediante il numero del candidato (uno 0 come preferenza di candidato implica *scheda bianca* mentre un valore non valido verrà inteso come *scheda nulla*) e archivia il voto espresso in forma anonima. Prevedete anche un meccanismo per cui uno studente che abbia già votato non può votare nuovamente (definite voi le strutture dati più appropriate per farlo).

Scrivete infine un metodo (che verrà invocato dopo ciascun voto) per lo spoglio automatico delle schede in cui il sistema stampa: il numero e le percentuali di votanti, il numero di schede bianche, il numero di schede nulle e il numero di voti presi da ciascuna candidato.

4) SIMULAZIONE DEL GIOCO DEL TRIS.

Regole del gioco: Si gioca in due su una scacchiera 3x3 inizialmente vuota. I giocatori alternativamente mettono rispettivamente uno O o un X in una casella vuota.
Vince chi riesce a completare un filotto di tre elementi per primo. Il filotto può essere orizzontale, verticale o diagonale. Se nessuno dei due riesce a fare un filotto e non ci sono più celle disponibili la partita è pari.

Problema: Scrivete un programma in grado di gestire partite a tris fra due giocatori umani o fra un giocatore e il computer. Lo svolgimento di una partita consiste nel chiamare alternativamente per i due giocatori un metodo, che potremmo chiamare *gioca*, e di verificare se la mossa è valida, se uno dei due giocatori ha vinto o se la partita è pari.

Per il giocatore umano il metodo *gioca* semplicemente chiederà all'utente dove inserire il segno.
Per il giocatore computer il metodo *gioca* potrebbe semplicemente inserire il segno in una qualsiasi posizione fra quelle disponibili.

Scrivete le classi necessarie allo scopo e utilizzatele in un programma di prova per testarne il corretto funzionamento (sia nel caso umano contro umano che umano contro computer).

Suggerimenti: Si potrebbero derivare i giocatori (umani e computer) da un'unica classe astratta giocatore. Probabilmente conviene anche definire una classe partita che gestisce lo svolgimento della partita stessa.

5) SIMULAZIONE PARTITA DI TENNIS

Il tennis è uno sport che vede opposti due giocatori (incontro singolare). Gli incontri hanno dei punteggi che si suddividono in partite, in inglese set; in base ai tipi di torneo gli incontri sono al meglio dei 3 o 5 set; una partita è costituita da 6 o più giochi, (games). Se i giocatori sono sul punteggio di 6-6 si continua finchè uno dei due giocatori vince con due giochi di scarto.

Scrivete le classi necessarie per la simulazione di una partita di singolare di tennis al meglio dei 3 set (la partita viene giocata da due giocatori e vince chi conquista 2 set su 3).

La classe **Giocatore** deve contenere la posizione in classifica del giocatore (int), nome e cognome (String).
La classe **Incontro** al suo interno contiene due oggetti della classe **Giocatore**, 3 oggetti della classe **Set** ed il metodo per effettuare la simulazione (quindi per giocare i set finchè uno dei due giocatori ne ha vinti 3).
La classe **Set** conterrà la sequenza dei risultati parziali del set (ad esempio: 0-0, 1-0, 1-1, 2-1, 3-1, 3-2, 4-2, 4-3, 5-3, 6-3) ed il metodo per effettuare la simulazione. La simulazione di un set avviene mediante la generazione di numeri casuali che stabiliscono se un gioco è stato vinto dal primo o dal secondo giocatore e dalla verifica dell'avvenuta vittoria da parte di un giocatore.

Il programma in particolare deve avere le seguenti funzionalità:

1) creare due oggetti **Giocatore**
2) creare un oggetto **Incontro** con i due giocatori appena creati
3) avviare la simulazione dell'incontro
    a. avviare la simulazione del primo set
    b. avviare la simulazione del secondo set
    c. se necessario avviare la simulazione del terzo set
4) stampare il risultato dettagliato di tutti i set giocati
5) stampare i dati del vincitore

Per esempio una esecuzione del programma potrebbe essere la seguente:

```
Inserisci dati giocatore 1: Mario Rossi 25
Inserisci dati giocatore 2: Luca Neri 32
Inizia la partita!
Simulazione set 1: vince Mario Rossi
Simulazione set 2: vince Luca Neri
Simulazione set 3: vince Luca Neri
Ha vinto la partita Luca Neri
Set 1: 0-0, 1-0, 2-0, 2-1, 3-1, 4-1, 5-1, 6-1
Set 2: 0-0, 1-0, 1-1, 1-2, 1-3, 2-3, 3-3, 3-4, 4-4, 5-4, 5-5, 5-6, 6-6, 6-7, 6-8
Set 3: 0-0, 0-1, 0-2, 0-3, 0-4, 1-4, 2-4, 2-5, 3-5, 3-6
```

6) GESTIONE CODA EVENTI

Nei sistemi operativi con interfaccia grafica gli eventi generati dall'utente (col mouse o con la tastiera) vengono trattati mediante una struttura dati di tipo coda, in cui gli elementi vengono inseriti in coda ed estratti dalla testa (modalità FIFO).
Scrivere le classi necessarie per la gestione di una coda di eventi.

Gli eventi possono essere di due tipi, **EventoTastiera** ed **EventoMouse** che derivano dalla classe astratta **Evento**. Ogni evento sarà caratterizzato da un identificativo (`int`) e da una descrizione testuale (`String`).
In aggiunta gli oggetti **EventoTastiera** avranno il codice del tasto premuto (`int`) e lo stato dei modificatori (`int`) mentre gli oggetti **EventoMouse** avranno la posizione del mouse sullo schermo (due `int`) e lo stato di pressione dei tasti del mouse (`int`). Implementate nelle classi **Evento**, **EventoTastiera** ed **EventoMouse** tutti i metodi che ritenete utili.

Scrivete un programma che crea una coda di eventi e, all'interno di un menu testuale, permette di creare e inserire un nuovo evento di tastiera, creare e inserire un nuovo evento del mouse e di prelevare e visualizzare un evento.