PROGRAMMAZIONE (9 CFU) – A.A. 2016-2017	ES. 06

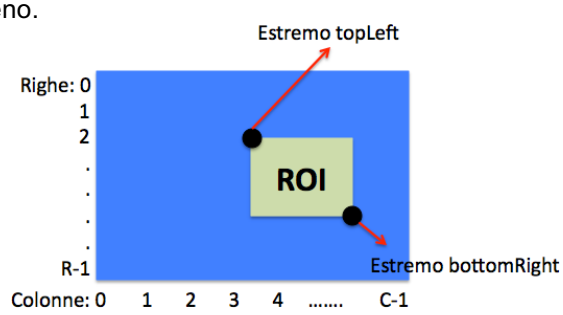
Inserire nel package **it.unipa.community.nomecognome.prg.n06.es0X** le seguenti applicazioni.

1) Si consideri la gerarchia di classi dell'esercizio 6 dell'esercitazione n05 per rappresentare: Veicolo, VeicoloAMotore, Ciclomotore, Automobile, Bicicletta (*se non avete accesso al codice dell'esercitazione 6, utilizzate la soluzione online*).
   Si modifichi la gerarchia in modo da implementare classi astratte. Se necessario, modificare la classe Veicolo in modo da prevedere un array di oggetti della classe Ruota. Modificare opportunamente le altre classi (se necessario) per istanziare il numero corretto di oggetti della classe Ruota.

2) Si modifichino ulteriormente le classi dell'esercizio precedente. Rendere abstract il metodo muovi() della classe Veicolo. Si supponga che i veicoliAMotore si muovano di moto uniformemente accelerato:
$$\vec{x} = \vec{x}_0 + \vec{v}\Delta t + \frac{1}{2}\vec{a}\Delta t^2$$
   mentre le Biciclette seguano la seguente legge di moto:
$$x = x_0 + v_x \Delta t + \frac{1}{2} a_x \Delta t^2$$
$$y = \cos(x)$$
   Si utilizzi una classe Main per testare le nuove classi.

3) Si scriva un'applicazione che consenta di simulare il lancio di un Dado e di una Moneta. Si utilizzi un'opportuna classe astratta per rappresentare le caratteristiche comuni delle due classi.
   Simulare, attraverso un opportuno metodo draw(), la visualizzazione del dado che rotola o della moneta che gira durante il lancio. In particolare, il metodo dado.draw() visualizza i numeri da 1 a 6 per un numero intero e casuale di volte (max 10) e i numeri da 1 fino al numero della faccia che è uscita *(per es. supponendo che dal lancio del dado si ottenga 3, una possibile visualizzazione potrebbe essere: "1 2 3 4 5 6 1 2 3 4 5 6 1 2 3").* Per la moneta si considerino i simboli T e C per indicare Testa e Croce.

4) Si supponga di voler elaborare segnali discreti 2D (immagini) rappresentabili attraverso matrici di numeri interi di dimensione R x C (R = numero di righe, C = numero di colonne).
   Si costruisca la classe Image che contiene le variabili d'istanza R, C, ed una matrice di interi data. Si utilizzi il costruttore per inizializzare tali matrici con dei valori generati da un oggetto della classe Random (con valori compresi tra 0 e 255). La classe Image deve consentire l'estrazione di regioni di interesse (ROI) rettangolari.
   Una ROI descrive una porzione rettangolare di un'immagine. Una ROI è definita attraverso due oggetti della classe Punto2D: il punto in alto a sinistra ed il punto in basso a destra. Ogni punto memorizza il valore di una riga (nelle y) e di una colonna (nelle x) sull'immagine. ROI possiede anche metodi per determinare il numero di righe e di colonne nella ROI (getHeight e getWidth rispettivamente).
   Inoltre, ROI possiede un metodo isValid() che valuta se il numero di righe e quello delle colonne nella ROI sono maggiori di 0 o meno.



   La classe Image contiene un metodo getRoiImage(Roi roi). Tale metodo restituisce un oggetto Image che memorizza la porzione dell'immagine descritta nella ROI. *(suggerimento: è possibile usare Arrays.copyOfRange per copiare una porzione di un array documentazione: https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html).*
   Testare le classi attraverso un programma che crea una immagine. Si estraggano dall'immagine 4 ROI ognuna che memorizzi un quarto dell'immagine.

5) Nel campo dell'elaborazione dei segnali (per esempio delle immagini), è comune utilizzare metodi di elaborazione numerica dei segnali per la rilevazione di fenomeni/eventi. Ogni evento può essere rilevato con una tecnica ben specifica (cioè un particolare detector). Si supponga di voler elaborare immagini 2D. Si iimplementi una gerarchia di classi che derivano da una classe astratta Detector. Un Detector ha un metodo astratto detect(Image image) che restituisce una ROI.
   Si considerino i seguenti detector: PedestrianDetector, CarDetector, FaceDetector. Ai fini di test, all'interno del metodo detect, generare ROI attraverso un opportuno oggetto della classe Random (nella realtà, ogni detector utilizza un algoritmo diverso per l'elaborazione dell'immagine e la generazione della ROI) sapendo che:
   - una ROI rappresentante un pedone (*pedestrian*) è un rettangolo con un'altezza pari almeno a 3 volte la larghezza;
   - una ROI rappresentante una macchina (*car*) è un rettangolo con una larghezza pari almeno a 2 volte l'altezza;
   - una ROI rappresentante una faccia è approssimativamente un quadrato (si consideri una tolleranza pari al 20% della dimensione più corta).

   Utilizzare queste informazioni in fase di generazione della ROI per generare delle regioni di interesse verosimili.

6) Nella gerarchia descritta nell'esercizio 5 dell'esercitazione 5 e relativa a Shape, Circle, Rectangle e Square, introdurre una nuova interfaccia Scalable che permetta di implementare un metodo scale(double factor). Il metodo consente di modificare le dimensioni dei lati/raggio di una figura in base al fattore di proporzionalità factor.

7) Ripetere l'esercizio 5 in modo che i tre i detector non sono più sottoclassi della classe astratta Detector, bensì implementino un'interfaccia comune DetectorInterface che dichiara un metodo ROI detect(Image).

NOTE PER COMPILAZIONE E TEST A RIGA DI COMANDO IN AMBIENTE LINUX:

Creare una cartella col proprio cognome sulla scrivania con all'interno le directory classes e src.
*Aprire una finestra di **terminale** e digitare:*
`cd Desktop/cognome/classes` *oppure* **`cd Scrivania/cognome/classes`**
*Creare i file sorgente con **gedit** e salvarli nelle sottodirectory di src corrispondenti al package.*
Digitare:
`javac -d . ../src/it/unipa/…/prg/n06/es0X/*.java` *(compila e genera il bytecode)*
`java it.unipa.community.….prg.n06.es0X.Main` *(esegue il bytecode sulla JVM)*