(A) CONOSCENZA TERMINOLOGICA

Dare una breve descrizione dei termini introdotti:

- Coordinate effettive
- Coordinate di schermo
- Unità di misura
- Piano di visualizzazione
- Classe **Point**
- Classe **Font**
- Classe **Image**
- Classe **Graphics**
- Classe **Canvas**
- Metodo **drawLine**()
- Metodo **drawRect**()
- Metodo **drawOval**()
- Metodo **fillRect**()
- Metodo **fillOval**()
- Metodo **drawImage**()
- Metodo **drawString**()
- Metodo **getImage**()
- Metodo **getSize**().**width**
- Metodo **Paint()**
- Metodo **getSize**().**height**

(B) CONOSCENZA E COMPETENZA

Rispondere alle seguenti domande producendo anche qualche esempio

B1) Conoscenza

1. A cosa serve la classe **Canvas**?
2. Quali sono gli attributi di un oggetto di classe **Font**?
3. Che differenza c'è tra le *coordinate effettive* e le *coordinate di schermo*?
4. Perché è necessario calcolare una *unità di misura* sugli assi?
5. A cosa serve il metodo *trasforma()*?

B2) Competenza

1. Qual è lo schema del metodo **paint()**?
2. Quali sono i parametri del metodo **drawString** ()?
3. Cosa indicano le istanze **getSize().width** e **getSize().height**?
4. Qual è la struttura di un'applicazione grafica
5. Quali classi servono complessivamente per rappresentare immagini?
6. Quali sono i *principali metodi* della classe **Graphics**?

(C) ESERCIZI DI COMPRENSIONE

1. Il …… è un particolare oggetto contenitore che viene utilizzato quando si vogliono realizzare applicazioni grafiche. Esso possiede un unico metodo, il metodo …….. che riceve come parametro un oggetto *g* di classe …….. Sull'oggetto *g* è possibile agire con i metodi di grafica.
2. Per realizzare un'applicazione grafica completa, occorre gestire un contenitore di tipo …….., i ……., ossia gli elementi grafici necessari, e gli …….. che questi ultimi sono in grado di generare.
3. All'interno del **Canvas**, l'oggetto *g* di classe **Graphics** consente di disegnare scritte ……, variabili per …….., …….. e ……. e di disegnare forme grafiche come …….., ……, …… e ……. anche utilizzando i colori.
4. Nel piano cartesiano, le coordinate sono espresse da numeri …….e prendono il nome di coordinate …… I loro valori vanno, generalmente, da - ∞ a + ∞; sullo schermo, invece, sono espresse da numeri …….. e prendono il nome di coordinate …….. I loro valori variano in base alla …….. del video utilizzato.
5. Il metodo getSize() consente di ricavare le dimensioni correnti dello schermo, espresse in ……, mediante gli attributi …….., che dà la misura della …….., e …….. che dà la misura della ……..
6. Per ciascuna delle proposizioni riportate, indicare se vera o falsa.

	Vero	Falso
Il **Canvas** è un componente che non può contenere controlli		
Le coordinate di schermo sono proporzionali a quelle effettive		
Le coordinate di schermo sono espresse da numeri reali		
Coordinate di schermo e coordinate effettive sono sinonimi		
Il metodo **paint(…)** appartiene alla classe **Graphics**		
Le scritte grafiche sono realizzate con un oggetto di classe **Graphics**		
Le forme grafiche sono realizzate con un oggetto di classe **Canvas**		
Il metodo **paint(…)** appartiene alla classe **Canvas**		

7. Scrivere nella colonna di destra, il prototipo del metodo opportuno che realizza quanto scritto a sinistra..

Per disegnare …	…si usa il metodo
… una linea…	
… un cerchio…	
… un ovale…	
… un rettangolo…	
… una scritta…	
… un arco…	

8. Scrivere nella colonna di destra, il prototipo del metodo opportuno che realizza quanto scritto nella prima colonna.

Per colorare …	**…si usa il metodo**
… un cerchio…	
… un ovale…	
… un rettangolo…	
… un arco…	

9. Scrivere lo schema di un'applicazione grafica che usi il *canvas*.

10. Scrivere le azioni che ordinatamente devono essere svolte per inserire in un frame un oggetto grafico *g* di classe *classeGrafica*.

11. Scrivere le istruzioni per stampare una stringa *str* in modalità grafica, dopo averne impostato il font ad "Arial" e il colore a *verde*..

12. Per ciascuno degli esercizi seguenti, completare le parti mancanti, correggere eventuali errori, determinare gli output prodotti e dare una breve descrizione dell'applicazione

```
class Retta extends Canvas
{  int x1, y1, x2, y2;
   public Retta (int a1, int b1, int a2, int b2)
   {        x1=…;
            y1=…;
            x2=…;
            y2=…;
   }
   public void paint(……..)
   {        setBackground (Color.gray);
   g.drawString ("P1(" + x1 + "," + y1 + ")", x1-50, y1-10);
   g.drawString ("P2(" + x2 + "," + y2 + ")", x2-50, y2+30);
   g.drawLine (x1, y1, x2, y2);
   }
}
```

13. Data la seguente classe *Triangolo*, completare le parti mancanti del costruttore con parametri e del metodo **paint**, tenendo presente che deve disegnare in verde un triangolo aventi per vertici i punti *t1*, *t2* e *t3*.

```
class DrawTriangle extends Canvas
{
            Point t1 = new Point(), t2=new Point(), t3=new Point();
            public DrawTriangle(Point p1, Point p2, Point p3)
            {
                …..
            }
            public void paint(Graphics g)
   {
            …..
   }
}
```

(D) ESERCIZI DI APPLICAZIONE

1. Scrivere un'applicazione che scriva una stringa in senso verticale
2. Modificare l'applicazione *DisegnaGrafico*, in modo da disegnare, sullo stesso *canvas*, due funzioni in colore diverso.
3. Scrivere un'applicazione che disegni un triangolo, note le coordinate dei vertici.
4. Modificare l'esercizio 3 in modo da disegnare un triangolo, conoscendo le misure dei lati.
5. Scrivere un'applicazione Java che, senza usare il metodo **drawOval()** disegni:
   a. una circonferenza, dato il centro (x, y) ed il raggio r;
   b. un'ellisse, dato il centro e i semiassi a e b.

6. Scrivere un'applicazione che disegni un poligono di N lati, di cui si conoscano le coordinate degli N vertici.
7. Modificare l'applicazione *DisegnaGrafico*, inserendo controlli per leggere i dati e pulsanti per disegnare il grafico e per l'uscita.
8. Disegnare il grafico di un'iperbole y = k / x, con k immesso dall'utente. Si tenga conto dell'insieme di definizione della funzione.
9. Disegnare il grafico della funzione y = log (x-2). Si tenga conto dell'insieme di definizione della funzione.
10. Disegnare un grafico che mostri l'andamento della temperatura in una data città, misurata ogni 3 ore e i cui valori sono riportati nella tabella a fianco.

Ora	**Temperatura (°C)**
0:0	15
3:0	17
6:0	19
9:0	20
12:0	26
15:0	25
18:0	20
21:0	17

11. Utilizzando le funzioni descritte in L2C1:
    - **public** trasforma(**double** x, **double** y)
    - **public void** asseX(**Graphics** g)
    - **public void** asseY(**Graphics** g)
    - **public void** origine(**Graphics** g)
    - **public static double** f (**double** x)

    implementare una classe che disegni il grafico di una funzione f a scelta tra:
    - **return Math**.**pow**(x, 2)+4*x+4; (valori di prova -10, 100, -10, 100)
    - **return Math**.**log**(x); (valori di prova -1, 10, -50, 10)
    - **return Math**.**sin**(x); (valori di prova -10, 10, -10,10)
    - **return Math**.**exp**(2); (valori di prova 0, 10, 0, 1)

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 1

Testo: Tramite le classi del *package* **AWT** (*Abstract Window Tool*), realizzare il logo delle olimpiadi, costituito dai classici cinque cerchi intersecati e di colore diverso.
Obiettivo: Utilizzo del package awt

1) Attivare l'ambiente di sviluppo (TextPad, Eclipse, ecc);
2) Creare un nuovo file e salvarlo come *Olimpiadi.java*. Scrivere il codice per implementare la classe *Olimpiadi*, che eredita da **Canvas** per poter usare il metodo **paint()**. Nella classe porre:
   a. nCerchi, intero;
   b. x, y, intero (posizioni origine);
   c. g di classe **Graphics**

   e come metodi:
   a. Olimpiadi(), costruttore senza parametri che inizializza *nCerchi* a 5;
   b. **void paint** (**Graphics** g) che inizializza *x* ed *y* entrambi a 10 e, tramite un ciclo con contatore *i* che va da 0 a 4, disegna i 5 cerchi. Tenere presente che:
      - l'ascissa *x* viene, per ogni cerchio, incrementata sempre di 130;
      - i 3 cerchi superiori (*i* = 1, 3, 5), hanno centro con y = 10;
      - i 2 cerchi inferiori (*i* = 2, 4), hanno centro in y=110;
      - il colore di ogni cerchio è impostato con il metodo
        void setColor (**int** i, **Graphics** g)
        descritto di seguito;
      - i cerchi hanno raggio 250.

```
void paint (Graphics g)
Inizio
    Intero i;
     x=10; y=10;
     Per i=0 a 4 fai
     Inizio
        Imposta il colore del cerchio;
        Disegna il cerchio con centro in (x, y) e raggio 250;
        Se i è pari y = 110;
           Altrimenti y = 10;
        x = x + 130;
     Fine
Fine
```

   c. **void** setColor (**int** c, **Graphics** g). Questo metodo, in base al valore di *c* imposterà un colore diverso, in base alla tabella a fianco.

c	Colore
0	black
1	red
2	blue
3	yellow
4	green

```
void setColor (int c, Graphics g)
Inizio
     Nel caso che c sia
     0: Imposta il colore black;
     1: Imposta il colore red;
     2: Imposta il colore blue;
     3: Imposta il colore yellow;
     4: Imposta il colore green;
Fine
```

3) Compilare il file *Olimpiadi.java* fino ad ottenere un codice corretto.
4) Creare un nuovo file e salvarlo con il nome *TestOliiipadi.java* e implementarlo in modo che svolga le seguenti operazioni:
   a. crei un oggetto *logo* di classe *Olimpiadi*;
   b. crei una finestra *f* di dimensioni (800, 400) posizionato in (200, 100);
   c. aggiunga l'oggetto *logo* alla finestra *f*;
   d. renda la finestra visibile.
5) Verificare la correttezza del disegno prodotto (v. Fig. 1)
6) Effettuare le seguenti modifiche:
   a. cambiare l'attributo *g* in: **Graphics2D** g2;
   b. nel metodo **paint()** prima del ciclo **for** assegnare: g2 = (**Graphics2D**) g;
   c. modificare tutti i precedenti riferimenti a *g* in *g2*;
   d. aggiungendo prima della chiamata a **drowOval()** il metodo
      g2.**setStroke**(**new BasicStroke**(6.0f));
7) Eseguendo di nuovo l'applicazione, facendo uso della classe **Graphics2D**, si nota che il metodo **setStroke()** consente di modificare lo spessore delle linee. Il disegno dovrebbe apparire ora come mostrato in Fig. 2.

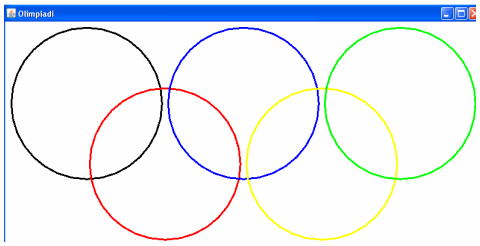

Fig. 1 - Logo olimpico

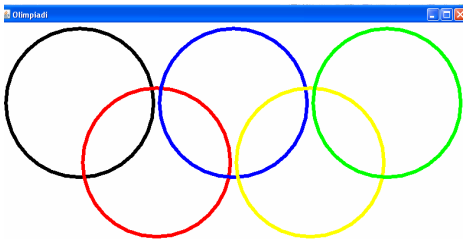

Fig. 2 - Logo olimpico con spessore variabile

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 2

Testo: Scrivere un'applicazione con interfaccia grafica che tracci il disegno di un tratto di curva geometrica, note le coordinate degli estremi (considerare solo il quadrante x>0, y>0). In particolare, considerare i casi di una retta e una parabola.
Obiettivo: utilizzo del **Canvas** per il disegno di curve geometriche.

1) Attivare l'ambiente di sviluppo (TextPad, Eclipse, ecc)

Costruzione della classe Retta

2) Creare un nuovo file e salvarlo come *Retta.java*. In questo file definire gli attributi x1, y1, x2, y2, tutti di tipo **int**.
3) Implementare i seguenti metodi:
   b. costruttore senza parametri;
   c. costruttore con parametri;
   d. metodo **paint()**:

```
public void paint(Graphics g)
{
        setBackground(Color.gray);
        g.drawString ("P1(" + x1 + "," + y1 + ")", x1-50, y1-10);
        g.drawString ("P2(" + x2 + "," + y2 + ")", x2-50, y2+30);
        g.drawLine (x1, y1, x2, y2);
}
```

Costruzione della classe Disegna

4) Scrivere, in un file separato, salvato con nome *Disegna.java*, il metodo **main()** per testare la classe creata, che deve:
   a. leggere da input (da finestra di comando o finestra grafica) le coordinate di due punti (estremi del segmento);
   b. creare un *frame f* in cui verrà inserito l'oggetto *r* di classe *Retta*;
   c. registrare un ascoltatore per la chiusura della finestra da casella di controllo;
   d. dimensionare la finestra a (400, 600);
   e. posizionare la finestra a (200, 200).
5) Testare l'applicazione con i valori: indicati a fianco

X1	Y1	X2	Y2
60	50	150	200
100	100	300	300
100	200	200	300
100	200	300	400

Costruzione della classe Parabola

6) Creare un nuovo file e salvarlo come *Parabola.java*. In questo file definire gli attributi x1, y1, x2, y2, tutti di tipo **int**.
7) Implementare i seguenti metodi:
   a. costruttore senza parametri;
   b. costruttore con parametri;
   c. metodo **paint()**:

```
public void paint(Graphics g)
{       int i;
        setBackground(Color.gray);
        g.drawString ("P1(" + x1 + "," + y1 + ")", x1-50,
y1-10);
        g.drawString ("P2(" + x2 + "," + y2 + ")", x2-50, y2+30);
        for (i=x1; i<=x2; i++)
          g..drawLine(i, (int)Math.pow(i, 2)/10, i+1, (int)Math.pow(i+1, 2)/10) ;
  }
```

X1	Y1	X2	Y2
0	100	70	200
5	50	80	100
5	100	200	200
5	100	200	120

8) Modificare l'applicazione *Disegna.java* in modo da sostituire l'oggetto di classe *Retta* con quello di classe *Parabola*.
9) Testare l'applicazione con i valori indicati a fianco:

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 3

Testo: scrivere un'applicazione con interfaccia grafica che tracci il disegno di una circonferenza, mediante l'uso delle coordinate polari.
Obiettivo: utilizzo del **Canvas** per il disegno di curve geometriche.

Dalla goniometria, si ha che un qualunque punto della circonferenza di centro (cx, cy) e raggio R ha coordinate:

x = cx + R * **cos** α; y = cy + R * **sin** α;

Considerando:

```
int xiniz=cx+raggio;  // ascissa iniziale
int yiniz=cy;         // ordinata iniziale
int xfin;             // ascissa finale
int yfin;             // ordinata finale
```

il disegno può essere realizzato mediante il metodo g.**drawLine** (xiniz, yiniz1, xfin, yfin), inserito in un ciclo in cui:

```
alfa+ = 0.0001;   // passo dell'angolo
xfin = cx + R * cos α;
yfin = cy - R * sin α;
xiniz = xfin;
yiniz = yfin;
```

Costruzione della classe Circonferenza

1) Creare un nuovo file e salvarlo come *Circonferenza.java*. In questo file compaiono 2 classi:
   a. Classe *DisegnaCirconferenza*
   b. Classe *Circonferenza*
2) Classe *DisegnaCirconferenza*
   In questa classe, creare un frame *f* , posizionarlo in (200, 200), e dimensionarlo a (600, 400). Successivamente, creare l'oggetto Circonferenza c = **new** Circonferenza (250, 200, 180) (che indica una circonferenza di centro in 250 e 200 e raggio180) e aggiungerlo al frame *f*. Rendere il frame *f* visibile.
3) Classe *Circonferenza*
   Questa classe usa il metodo **paint()** per disegnare la circonferenza ed è costruita come segue:

```
class Circonferenza extends Canvas
{   int cx, cy;       // coordinate del centro
    int raggio;       // valore del raggio
    public Circonferenza (int x, int y, int r)          // costruttore
    {   cx=x;   cy=y;
        raggio=r;
    }
    public void paint (Graphics g)
    {   int xiniz=cx+raggio;  // ascissa iniziale
        int yiniz=cy;         // ordinata iniziale
        int xfin;             // ascissa finale
        int yfin;             // ordinata finale
        double p=Math.PI;   // pi greco
        double alfa=0;        // angolo variabile da 0 a 2*p
        do
        {       alfa+=0.0001;     // calcolo punto successivo
                xfin=cx + (int)(raggio * Math.cos(alfa));
                yfin=cy - (int)(raggio * Math.sin(alfa));
                g.drawLine(xiniz, yiniz, xfin, yfin);   // disegna segmento
                xiniz = xfin;                           // il successivo diventa l'attuale
                yiniz = yfin;
        } while (alfa <= 2*p);
    }
}
```

4) Testare l'applicazione per verificarne il corretto funzionamento

(E) ESERCITAZIONI PRATICHE
Esercitazione n. 4

Testo: scrivere un'applicazione con interfaccia grafica che tracci il disegno di un istogramma, che rappresenti la distribuzione di valori interi contenuti in un array. .
Obiettivo: utilizzo del **Canvas** per il disegno di curve geometriche, creazione ed utilizzo del fattore di scala, disegno di rettangoli colorati.

Progettare e realizzare un'applicazione Java che tracci un istogramma sul piano, relativo ad un campione numerico i cui dati vengono memorizzati in un array.
L'applicazione deve prevedere una finestra ripartita in due aree: quella centrale, un canvas, è dedicata alla tracciatura dell'istogramma, mentre l'area inferiore contiene almeno i seguenti pulsanti:

- Carica dati
- Annulla dati
- Traccia istogramma
- Esci

Prevedere le seguenti classi:

- una classe *gestoreF* per l'evento di chiusura della finestra;
- una classe *gestoreP* che funge da handler degli eventi provenienti dai vari pulsanti;
- una classe *Istogramma* che contenga come attributi:
  - **int** x, y; // origine del grafico
  - **int** bf, hf; // finestra di visualizzazione
  - **int** v[]; // valori da rappresentare
  - **int** n; // ampiezza del campione
  - **double** c; // fattore di scala
  - **int** d; // interdistanza
  - **int** b; // base rettangoli
  - **Color** col; // colore rettangoli

  e come metodi:
  - costruttore **public** *Istogramma*(**int** x, **int** y, **int** bf, **int** hf, **int** n, **Color** col)
  - **public int** *vmax*() per calcolaree il massimo del campione;
  - **public void** *cancellaValori*(), che azzera il vettore del campione;
  - **public void** *caricaValori*(), per caricare i valori del campione;
  - **public void paint** (**Graphics** g) che contiene il ciclo di tracciatura dei rettangoli.
- una classe *TestIstogramma*, che crei un oggetto di classe *Istogramma* e predisponga l'interfaccia grafica completa.

Al termine, sono richiesti:

- la codifica del software richiesto;
- la documentazione con tecnologia **javadoc** (pagine web);
- la dimostrazione del corretto funzionamento