# Java 2D: desenarea si crearea elementelor grafice 2D

Java 2D API este libraria de baza care, impreuna cu Swing API, permite crearea de widget-uri non-standard, prin afisarea de imagini si figuri geometrice 2D.

Java 2D API ofera posibilitatea de a:

- desena figuri geometrice si completarea lor cu o culoare, gradienti sau texturi
- desenare de text
- desenare de imagini
- aplicarea unor operatii complexe precum compunere si transformari pe parcursul oricarei operatii de redare de mai sus.

API-ul Java 2D asigura toate aceste capabilitati pe baza extinderii framework-ului AWT (Abstract Windowing Toolkit). Obiectele Java 2D exista intr-un spatiu plan denumit: spatiu de coordonate utilizator sau pe scurt spatiu utilizator – *user space.*

In momentul redarii – *rendering phase* – obiectelor pe ecran sau la nivelul unei imprimante coordonatele din spatiul utilizator sunt tranformate in coordonate specifice dispozitivului – *device space coordinates.*

Clasele de baza in Java 2D:

- `Graphics`
- `Graphics2D`

Prin intermediul claselor din Java 2D API se asigura urmatoarele capabilitati:

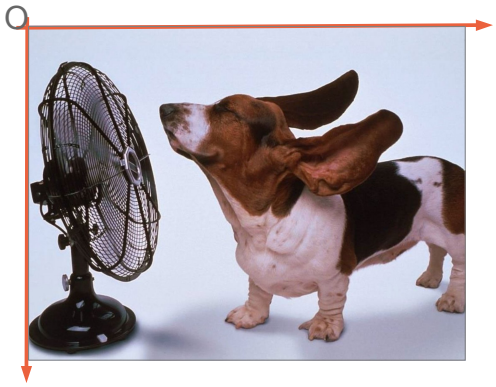
- un mode uniform de redare pentru dispositive de afisare si printare
- un set amplu de primitive geometrice
- mecanisme de detectare a evenimentelor generate pe figure, text sau imagini
- un modele de compunere si organizare care permite controlul précis asupra modului de redare a obiectelor grafice suprapuse
- control asupra redarii culorilor
- suport pentru printarea documentelor complexe

## 1. Spatii de coordonate

Java 2D API administreaza doua spatii de coordonate: user space si device space.

User space reprezinta zona de coordonate unde se definesc primitivele grafice si elementele ce pot fi realizate utilizand Java 2D API. User space este independent de dispozitivul care va realize redarea elementelor grafice realizate.

Device space este spatiul de coordonate specific unui dispozitiv de iesire precum monitorul sau imprimanta. Inainte ca elementele grafice sa fie redate la nivelul unui astfel de dispozitiv se realizeaza o transformare din user space in device space. Pentru a se putea realiza correct aceasta transformare se defineste o origine de baza pentru aria de desenare a componentei grafice si anume: coltul din stanga-sus. Coordonata pe axa Ox creste catre dreapta in timp ce coordonata pe axa Oy creste in jos.



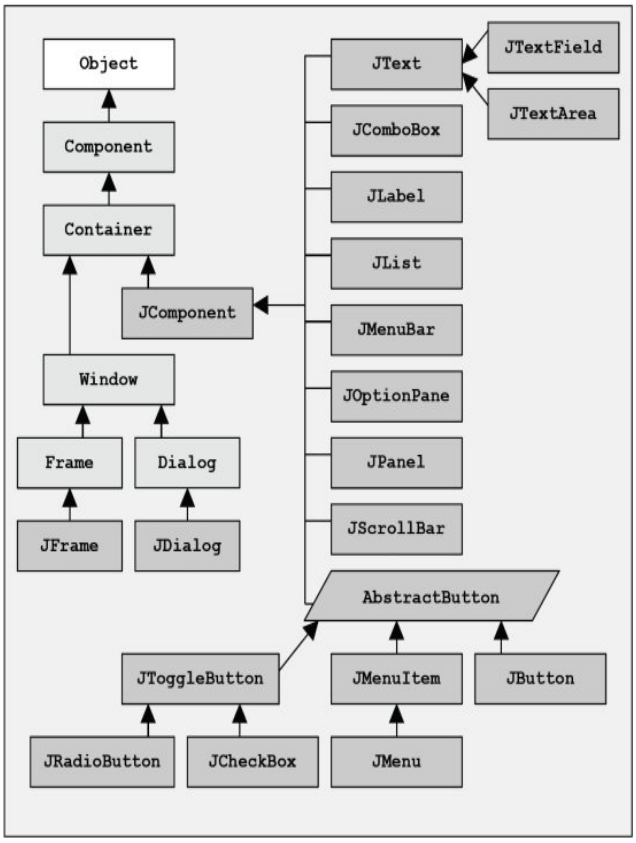
Transformarea din user space in device space se realizeaza automat in timpul operatiei de randare grafica.

De regula coordonatele si dimensiunile se definesc folosind intregi dar exista support si pentru cazurile speciale in care se cere o precizie deosebita – caz in care se pot folosi numere in virgule mobile float sau double.

**2. Redarea grafica in Java 2D**

API-ul Java 2D asigura, la nivelul unui set variat de dispozitivem un model uniform de redare a elementelor grafice. Astfel din perspectiva dezvoltarii, la nivelul aplicatiei, se asigura acelasi mecanism de redare indiferent de dispozitivul care va prelua si reda acele elemente grafice.

Din structura ierarhica de mai jos se observa ca orice componenta grafica (fie ea partea a Swing API sau a Java2D API) extinde java.awt.Component.




[Java Course – Ch.06]

La nivelul clasei Component sunt definite o serie de metode utilizate in redarea entitatii grafice create. Astfel, iin contextul redarii grafice, ori de cate ori trebuie evaluata si afisata o parte grafica a unei entitati de tip Component (ce extinde Component) se apeleaza una din metodele **paint(..)** sau **update(..)** sau variante supraincarcate ale acestora. Acest tip de metode primesc ca parametru contextul grafic in care se realizeaza redarea, context reprezentat prin obiecte de tip **Graphics** sau o extensie **Graphics2D**.

Pe langa redarea stricat a elementelor grafice, entitatile de tip **Graphics2D** permit metode care afiseaza o anumita forma geometrica sau metode care afecteaza redarea grafica:

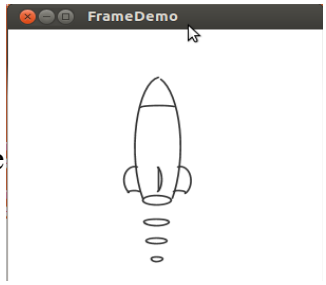
- desenarea conturului oricarei primitive geometrice (metode de tip **draw**)
- desenarea oricarei primitive geometrice (metode de tip **fill**)
- redarea unui text (metode de tip **drawString**)
- redarea unor imagini (metode de tip **drawImage**)
- specificarea aspectului liniei de contur – *stroke*
- specificarea modului de imbinare a capetelor de contur
- specificarea unei zone restranse asupra careia sa se aplice elemente si mecanisme de redare/transformare grafica
- metode de transformare: translatie, rotatie, scalare, forfecare
- definirea culorilr si a tiparelor de umplere
- definirea metodelor de compunere intr-un spatiu 2D a mai multor obiect (efect de colaj)

Metoda **Component.paint(Graphics g)** specifica oricarei componente grafice defineste ca argument un elemnt de tip **Graphics**. Pentru a accesa toate functionalitatile enumerate mai sus se obtine un element **Graphics2D** printr-o operatei de conversie explicita:

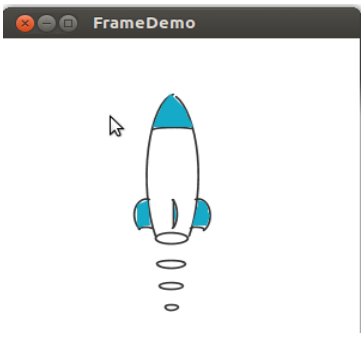
```
@Override
public void paint(Graphics g) {
	Graphics2D g2D = (Graphics2D) g;
}
```

Urmariti API – ul pentru o vedere mai ampla asupra componetelor `Graphics` si `Graphics2D`.
Este indicata utilizarea contextului grafica reprezentat printr-un obiect Graphics2D deoarece acesta asigura un set semnificativ de caracteristici :

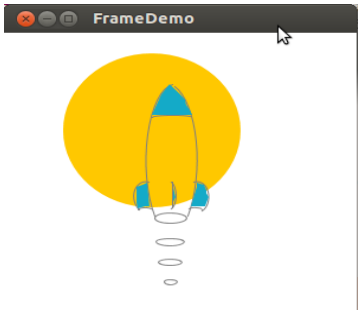
- contur – *pen attribute* – caracteristica aplicata pentru a specifica tipul si grosimea liniilor cu care se deseneaza un contur (linie punctata, grosime, modul de imbinare al liniilor)



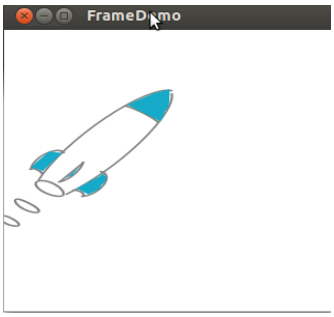
- completare/umplere - *fill attribute* – caracteristica aplicata la nivelul ocuparii cu un tipar, gradient sau culoare a interiorului unei primitive geometrice




- compunere – *compositing attribute* – utilizata pentru specificarea oridinii de afisare si suprapunere a obiectelor grafice




- transformare este caracteristica aplicata pentru a realiza tranzitia intre spatiul de coordonate al utilizatorului si cel al dispozitivului



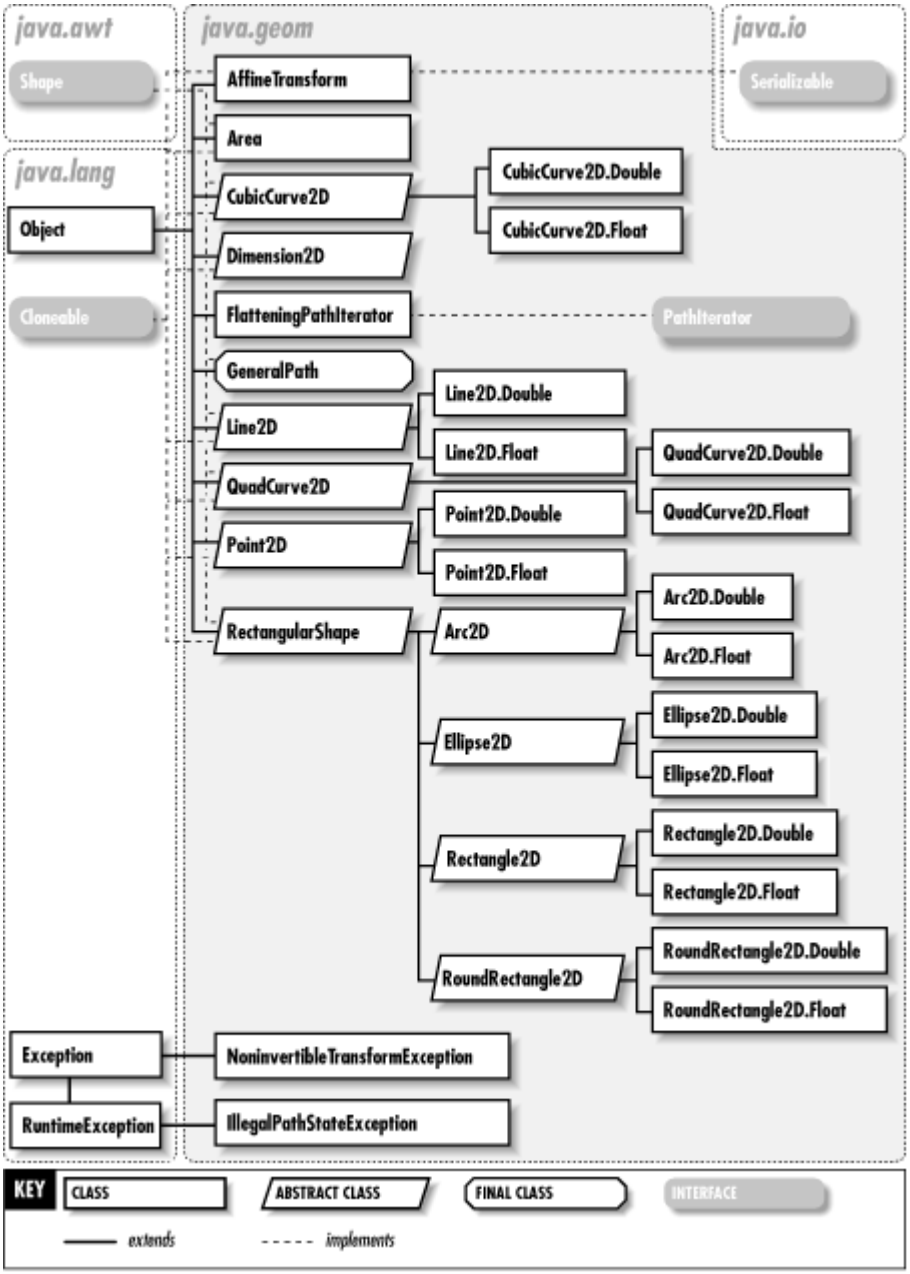

- zona de intersectie – *clip* – este o forma geometrica utilizata pentru a restrange aria unde se aplica restul caracteristicilor
- font-ul – caracteristica aplicata pentru redarea textului
- indicativi de redare – *rendering* hints – au fost introdusi ca mecanism de parametrizare a mecansimului de redare pentru a permite o definire a raportului calitate/viteza in functie de caracteristicile specifice aplicatiilor grafice realizate.

Pentru aprofundarea elementelor de mai sus se poate parcurge urmatorul document Advanced Topics in Java2D.

**4. Primitive geometrice**

API-ul Java 2D permite reprezentarea unui set standard de elemente grafice precum: linii, puncte, poligoane, elipse, curbe, arce. Pachetul java.awt.geom defineste un set cu cele mai uzuale primitive geometrice.

Elementele geografice definite la nivelul pachetului java.awt.geom implementeaza interfata Shape utilizata pentru a representa o forma geometrica care prezinta un contur si un spatiu geometric delimitat de acest contur.




Pachetul java.awt.geom [Java Foundation Classes in a nutshell]

Interfata Shape este o componenta importanta a arhitecturii API-ului Java 2D. Shape defineste metodele necesare pentru operatii asupra formelor geometrice.

public abstract interface **Shape** {
//
*Public Instance Methods*

public abstract boolean **contains** (java.awt.geom.Point2D *p*);

public abstract boolean **contains** (java.awt.geom.Rectangle2D *r*);

public abstract boolean **contains** (double *x*, double *y*);

public abstract boolean **contains** (double *x*, double *y*, double *w*, double *h*);

public abstract Rectangle **getBounds** ();

public abstract java.awt.geom.Rectangle2D **getBounds2D** ();

public abstract
java.awt.geom.PathIterator **getPathIterator** (java.awt.geom.AffineTransform *at*);

public abstract
java.awt.geom.PathIterator **getPathIterator** (java.awt.geom.AffineTransform *at*,
double *flatness*);

public abstract boolean **intersects** (java.awt.geom.Rectangle2D *r*);

public abstract boolean **intersects** (double *x*, double *y*, double *w*, double *h*);
}

Pe langa clase din java.awt.geom exista si alte tipuri care implementeaza Shape (e.g. java.awt.Polygon si java.awt.Rectangle). Pentru detalii privind metodele declarate in interfata Shape se va consulta documentatia API-ului corespunzator Java.

Dintre clasele care implementeaza Shape: `Polygon, Rectangle, java.awt.geom.Area, java.awt.geom.CubicCurve2D, java.awt.geom.GeneralPath, java.awt.geom.Line2D, java.awt.geom.QuadCurve2D, java.awt.geom.RectangularShape`

API-ul Java 2D defineste o serie de tipuri si mecanisme pentru descrierea, reprezentarea, abstractuizarea si redarea grafica a diverselor forme geometrice ce pot fi descrise intr-un spatiu 2D. Printre acestea se pot mentiona notiunile de: punct geometric, linie, forme rectangulare, curbe si curbe cuadratice, poligoane, arii geometrice.

Pentru detalii privind dezvoltarea in directia desenarii de contururi geometrice si umplerea acestora cu tipare grafice se poate consulta continutul de la urmatoarea adresa Working with Geometry.

**Punct geometric**

In Java2D un punct geometric este reprezentat prin tuplul (x,y) care reprezinta o locatie in spatiul de coordonate bidimensional. Clasa care defineste un astfel de obiect este Point2D si reprezinta un punct in sens geometric: nu are arie (spre deosebite de un pixel), nu detine informatie de culoare si nu poate fi redat grafic. Un ansamblu de puncte este utilizat pentru a crea si reprezenta alte forme geometrice. Points are used to create other shapes.

La aceasta adresa poate fi consultat API-ul Point2D.

**Linii**

Similar cu Point2D exista o clasa care introduce notiunea de linie geometrica prin clasa Line2D.

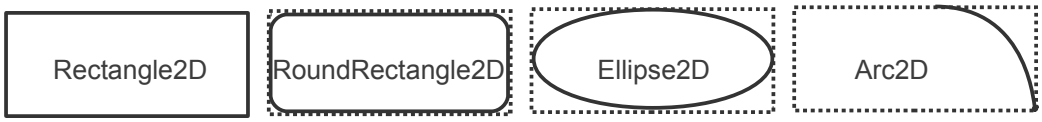
**Forme rectangulare**

Pentru reprezentarea abstracta a notiunii de forma rectangulara Java 2D introduce clasa RectangularShape ce defineste metodele necesare definirii si interactiunii obiectelor Shape care pot fi inlcuse intr-un contur rectangular.

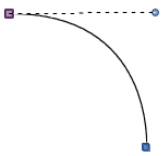
Geometric, un obiect RectangularShape poate fi extrapolat dintr-un obiect rectangular care inchide complet conturul unui obiect Shape.

RectangularShape este extinsa de clasele Rectangle2D, Ellipse 2D, RoundRectangle2D si Arc2D.

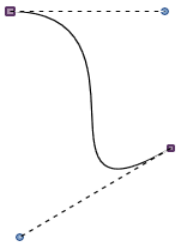
RectangularShape API.




**Curbe cuadratice**

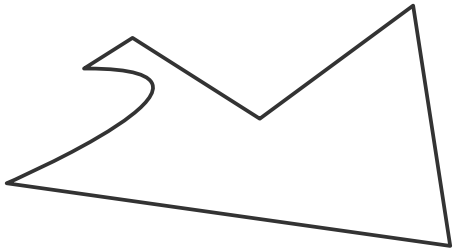


Java 2D introduce clasa QuadCurve2D care permite definirea si redarea unor segmente de curba quadratica definite pe baza a doua puncte de delimitare si a unui punct de control.



Similar, clasa CubicCurve2D permite crearea segmentelor de curba cubic parametricaza (i.e. doua puncte pentru delimitare si doua puncte de control).

**Poligoane neregulate si forme geometrice arbitrare**



Pentru definirea unei figure geometrice oarecare se poate folosi clasa GeneralPath prin definirea unui set de pozitii successive care se afla amplasate pe conturul figurii. Aceste puncte sunt conectate prin segmente de linie, curbe cuadratice sau cubice.

**Arii**

Clasa Area defineste mecanisme care permit operatii logice precum reuniune, intersectie si diferenta peste oricare doua obiecte de tip Shape.

Mai multe informatii pot fi consultate la adresa: http://docstore.mik.ua/orelly/java-ent/jfc/.

**5. Imagini**

Java 2D API permite redarea imagnilor grafice de diverse tipuri (PNG, GIF, JPG).

O imagine este descrisa printr-o matrice bidimensionala de pixeli. Fiecare pixel reprezinta informatia de culoare la pozitia respectiva in imagine.

Clasa de baza utilizata in reprezentarea obiectelor de tip imagini este `java.awt.image.BufferedImage`.

O aplicatie Java 2D poate crea un obiect `BufferedImage` sau poate sa genereze un astfel de obiect pe baza unui fisier imagini de format PNG sau GIF.

Prin apeluri Java2D Graphics/2D se pot desena si reda figuri gemoetrice (sau alte imagini) peste obiectul imagine redat.

Java2D asigura si operatii de filtrare asupra obiectelor de tip imagine: un filtru de tip `ConvolveOp` poate fi utilizat pentru a induce efectul de ceata sau claritate a imaginii.

Pentru detalii suplimetare consultati tutorialul Working with Images.