РЕАЛІЗАЦІЯ АНІМАЦІЇ
І
ВІДТВОРЕННЯ ЗВУКУ

ОЧІКУВАННЯ ЗАВАНТАЖЕННЯ ЗОБРАЖЕНЬ

Необхідність введення класів очікування завантаження зображення:

- Завантаження зображень - **тривалий процес** (зображення завантажуються навігатором в окремій задачі).
- Якщо файл зображення має **велику довжину**, воно з'являтиметься у вікні поступово у міру завантаження (може малювати відразу декілька зображень в різних місцях вікна або показувати їх по черзі в одному і тому ж місці для анімації).

Доцільно завантажити зображення повністю, а лише потім виконувати малювання, щоб зображення з'явилося на екрані відразу.

Два способи визначити, коли зображення завантажене повністю:

1) використання класу ***MediaTracker***;

2) перевизначення методів інтерфейсу ***ImageObserver***.

ЗАСТОСУВАННЯ КЛАСУ *MediaTracker*

Створення об'єкту класу

```
MediaTracker mt;
mt = new MediaTracker(this);
```

// посилання на компонент, для якого необхідно
// відстежувати завантаження зображень.

!!! Тобто, ми просто готуємо “диспетчера” і скажемо йому, за ким (за якими картинками) він далі буде спостерігати.

Додавання зображень в об'єкт класу MediaTracker

addImage()

Наприклад,

```
Image img1, img2, img3, img4;
        // спочатку треба створити всі необхідні
    зображення
img1 = getImage(getCodeBase(), "pic1.gif");
img2 = getImage(getCodeBase(), "pic2.gif");
img3 = getImage(getCodeBase(), "pic3.gif");
img4 = getImage(getCodeBase(), "pic4.gif");
          // далі додати їх в об'єкт MediaTracker
mt.addImage(img1, 0);
mt.addImage(img2, 0);
mt.addImage(img3, 1);
mt.addImage(img4, 1);
          // другий параметр – номер групи зображень
```

Очікування завантаження доданих зображень

(щоб переконатися, що всі зображення завантажені)

```
void waitForAll()
```

Наприклад,

```
try  {  mt.waitForAll();
        . . .
      }
catch (InterruptedException ex)
      {  . . . }
boolean  waitForAll (long ms)

boolean  waitForID (int id, long ms);
```

// повертають *true*, якщо за вказаний час всі зображення були
// успішно завантажені

Перевірка завершення завантаження

```
boolean checkAll(boolean load);
boolean checkID(int id);
boolean checkID(int id, boolean load);
```

// Якщо *load = true*, завантаження зображень ініціюється,
// якщо *false* - тільки перевірка поточного стану завантаження.

Стеження за процесом завантаження

```
int statusAll (boolean load);

int statusID (int id, boolean load);
```

Біті стану	Опис
LOADING	Один або декілька відстежуваних файлів продовжують завантажуватися
ABORTED	Завантаження одного або декількох файлів було перерване
ERRORED	При завантаженні одного або декількох файлів відбулася помилка
COMPLETE	Завантаження всіх відстежуваних файлів відбулося повністю і успішно

Обробка помилок

```
boolean   isErrorAny();
boolean   isErrorID (int id);
Object[]   getErrorsAny ();
Object[]   getErrorsID (int id);
```

ЗАСТОСУВАННЯ КЛАСУ *ImageObserver*

- Використовується для відстежування процесу завантаження і перемальовування зображень і інших компонент, розташованих усередині даного компоненту.

- Клас Component реалізує його. Слід перевизначити метод

```
boolean  imageUpdate (Image img,
   int flags, int x, int y, int w, int h);
```

flags – відображає стан бітів зображення

flags – відображає стан бітів зображення

WIDTH=0	Зображення завантажене настільки, що стала доступна його ширина (w).
HEIGHT=2	Аналогічно попередньому, але для висоти зображення (h).
PROPERTIES=4	Стали доступні властивості, які можна одержати методом getProperty класу Image.
SOMEBITS=8	Стали доступні біти зображення для малювання в масштабі (x, y, h і w)
FRAMEBITS=16	Завантажений черговий фрейм зображення, що складається з декількох фреймів. x, y, h і w слід ігнорувати
ALLBITS=32	Зображення завантажене повністю. Параметри x, y, h і w слід ігнорувати
ERROR=64	При завантаженні відбулася помилка
ABORT=128	Завантаження зображення було перерване або відмінене

- Додаток (аплет) повинен передати останньому параметру метода *drawImage()* посилання на інтерфейс *ImageObserver*, який буде відстежувати процес завантаження:

```
g.drawImage (Img, x, y, w, h, this);
```

Приклад перевантаженого методу *imageUpdate()*

```
public boolean imageUpdate (Image im,int flags,
  int x, int y, int w, int h)
{
      // Перевіряємо, чи все біти зображення завантажені
  fAllLoaded = ((flags & ALLBITS) != 0);
  // Якщо все, перемальовуємо вікно
  if(fAllLoaded) repaint();
      // Якщо всі біти завантажені,
      // подальші виклики цього методу не потрібні
  return (!fAllLoaded);
}
```

УСУНЕННЯ МЕРЕХТІННЯ

- **Перевантажити метод *update()***. Процес перемальовування вікна аплета займає більше часу, ніж період розгортання зображення по вертикалі на екрані монітора. Тому зображення з'являється у вікні в декілька прийомів

- Замість того, щоб періодично викликати метод repaint(), можна малювати в потоці, тобто **малювати в методі *run()***.

- Метод **подвійної буферизації**: зображення малюється не у вікні, а готується в оперативній пам'яті і потім, уже готове, виводиться у вікно, - процес підготовки не видимий користувачу.

Реалізація подвійної буферизації

```
……
 Image Img;
 Graphics grImg;
 Dimension  Dim = null;

public void paint(Graphics g)
 {
  if (Img != null) g.drawImage(Img,0,0,null);
 }
public void update(Graphics g)
{        // Визначаємо розміри вікна аплета
     Dimension dim = getSize();
     int W = dim.width;
     int H = dim.height;
```

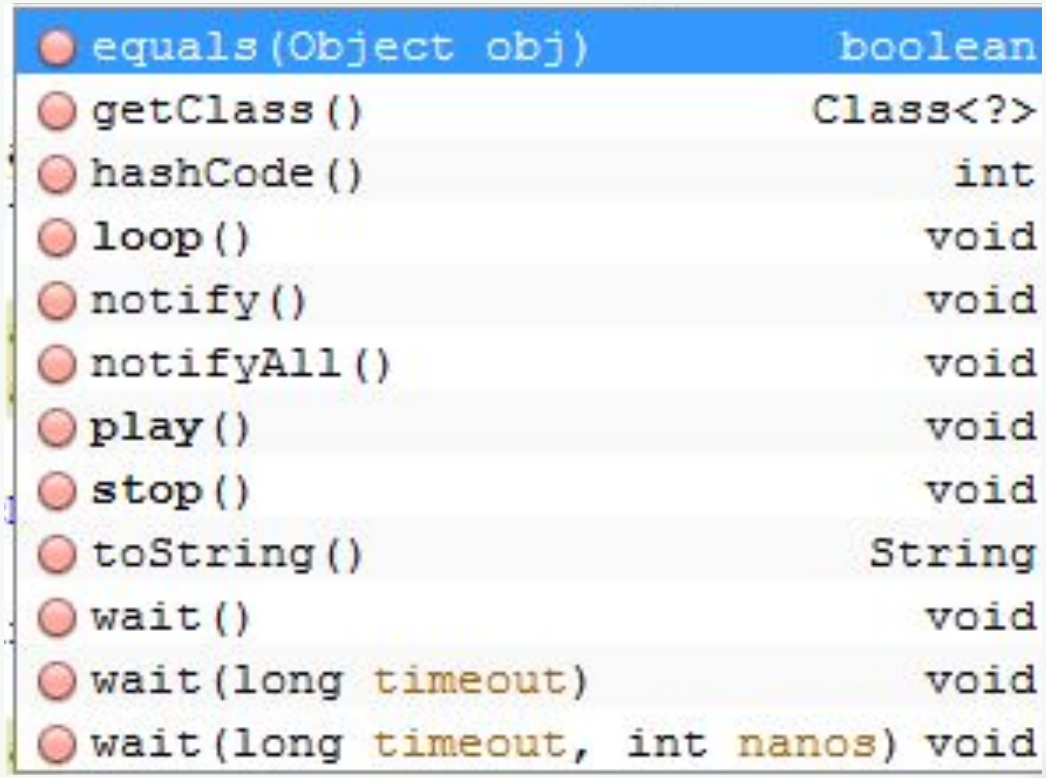
```
  // якщо зображення немає, то створюємо пусте розміром з
  вікно
  if ((Dim==null)||(Dim.width!=W)||(Dim.height!=H))
   {  Dim = new Dimension(applW,applH);
      Img = createImage(applW, applH);
      grImg = Img.getGraphics();
   }
  Color fg = getForeground(); // беремо колір тексту
  Color bg = getBackground(); // беремо колір фону
  grImg.setColor(bg); // встановлюємо такий фон в  памяті
  grImg.fillRect(0,0,imgDim.width,imgDim.height);
  grImg.setColor(fg); // встановлюємо колір малювання
  grImg.drawString("Hello",pos,40);// малюємо в пам'яті
  position--;
  if (position <= 0)
        position = size().width;
  paint(g);
}
```

ВІДТВОРЕННЯ ЗВУКУ

```
AudioClip ac;

ac=Applet.newAudioClip(new
                URL("file:spacemusic.au"));
```

!!! Обов'язкова обробка виключних ситуацій.



Питання?