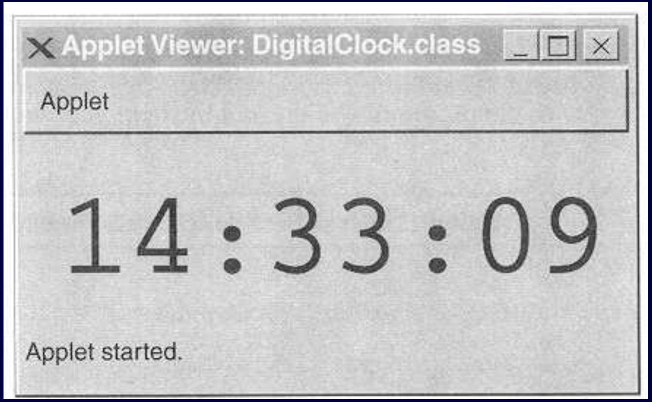
# 4. A Simple Animation Applet

- Java Programming – different from conventional
  - 1) object-oriented 2) framework-based…
- Framework-Based Programming
  - framework
    - provides the basic structure and utilities for applications
    - allow the application development effort to be reduced significantly.
    - extendable and flexible and hence can accommodate a broad range of application requirement and functionalities.
    - conventions and styles of framework must be followed
    - applications do not have full control of the system.

- inversion of control
  - top level of the system usually resides in the framework.
  - Applications must cooperate with the framework
- Interaction Styles : The Way in which Java Programs interact with users
  - 1) Active : run actively without input or intervention from the user ex) animation programs – Example 4.1
  - 2) Reactive : perform tasks in reaction to user input
    - user input : key strokes, mouse clicks, menu selections
    - ex) Example Chap 9.
  - 3) Hybrid : function bye themselves and also react to user input
    - ex) example Chap 8 [p. 305]

# Ex. 4.1. A Digital Clock Applet – Initial Version

- Example 4.1 A Digital Clock Applet – The Initial Version
  - for animation applets
  - Figure4.3 The digital clock applet



```
14:33:09
```

Applet started.

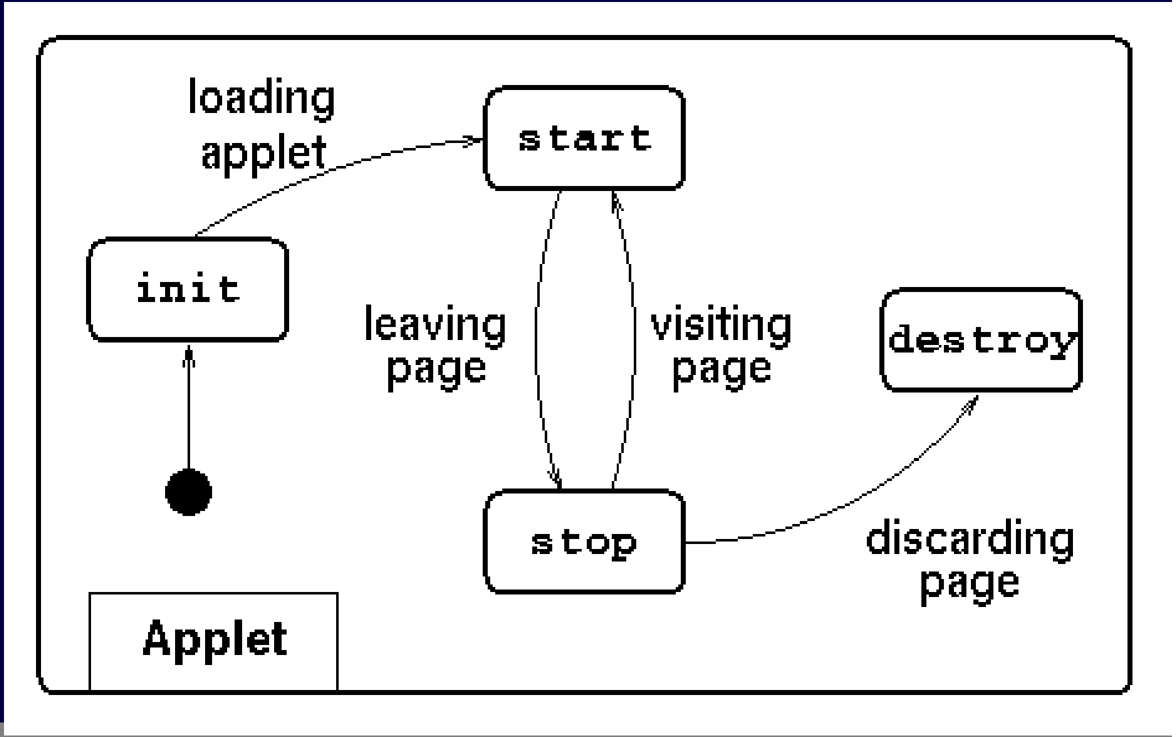
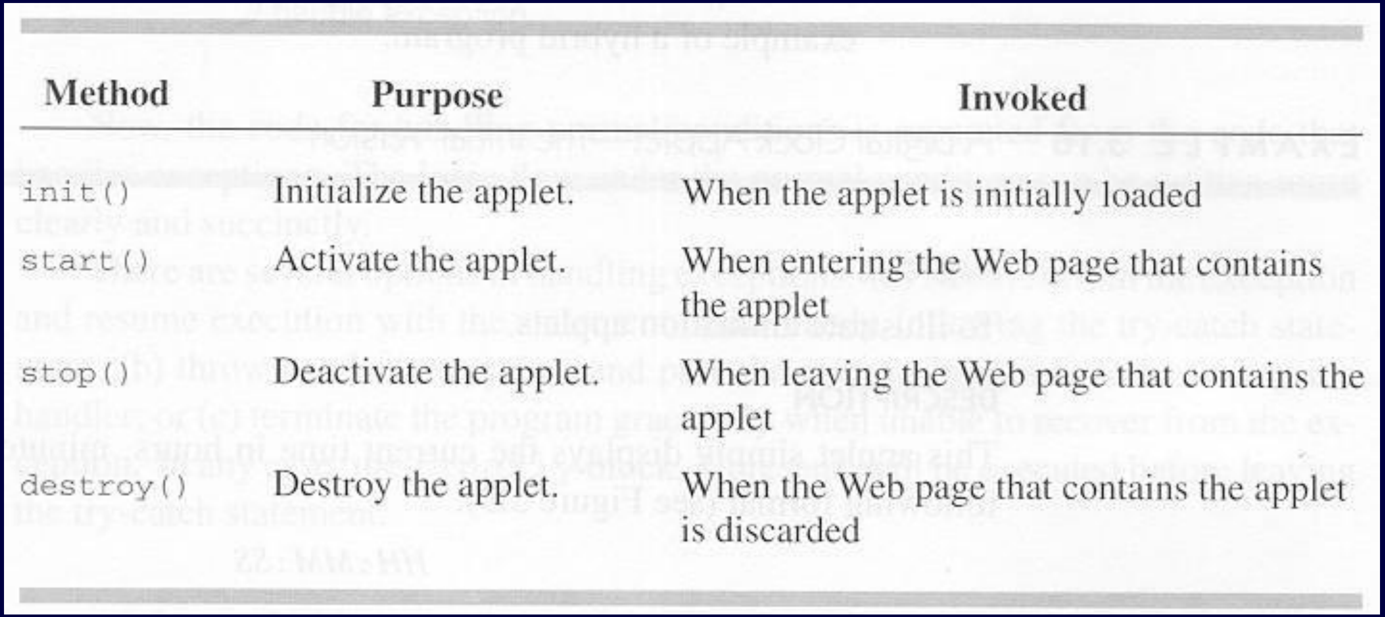


Fig 4.1 The Life Cycle of applets

## Ex. 4.1. A Digital Clock Applet – Initial Version

- Methods of applets



| Method | Purpose | Invoked |
|---|---|---|
| `init()` | Initialize the applet. | When the applet is initially loaded |
| `start()` | Activate the applet. | When entering the Web page that contains the applet |
| `stop()` | Deactivate the applet. | When leaving the Web page that contains the applet |
| `destroy()` | Destroy the applet. | When the Web page that contains the applet is discarded |

- Digital Clock Applet
  - Overriding three of methods(Applets) : init(), start(), stop()
  - define two other methods : paint(), run()

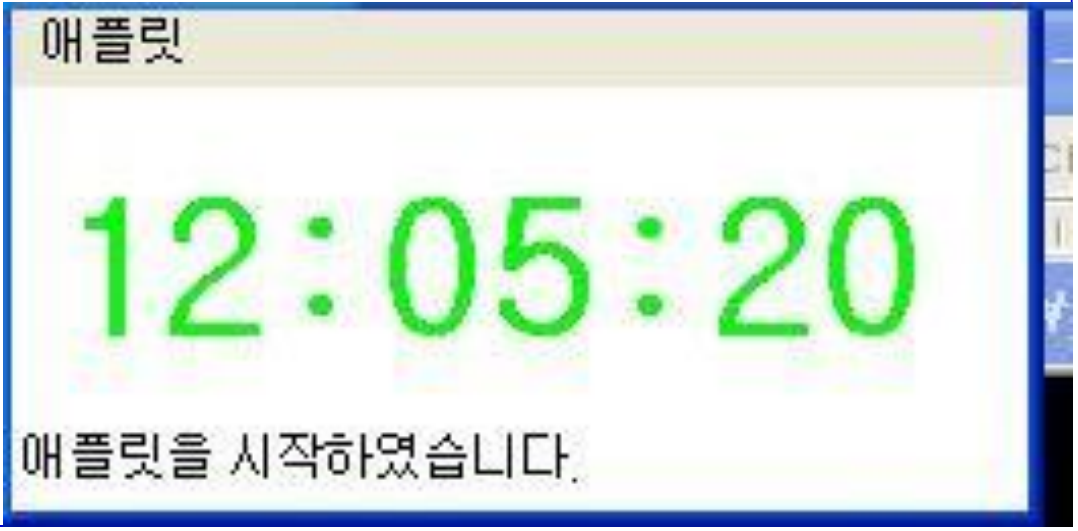
## Digital Clock (초기버젼)

```
// Example 4.1. A Digital Clock Applet - The Initial Version


import java.awt.*;
import java.util.Calendar;

/**
 *  This is an applet that displays the time in the following format:
 *        HH:MM:SS
 */

 // must be a subclass of java.applet.Applet
public class DigitalClock
  extends java.applet.Applet implements Runnable {

  protected Thread clockThread = null;
  protected Font font = new Font("Monospaced", Font.BOLD, 48);
  protected Color color = Color.green;
```

```
public void start() {
   if (clockThread == null) {  clockTh
     read = new Thread(this);  clockT
     hread.start();
   }
 }
 // it is important to deactive the applet by killing the animation thread
 // otherwise, the applet would keep running and consuming CPU and
memory resources
 //     after you leave the web page that contains the applets.

 public void stop() {
   clockThread = null;
 }

// run() : the main body of the thread.
 // analogous to the main() method of an application class.
 public void run() {
   while (Thread.currentThread()  ==  clockThread) {
     repaint();
     try {
            Thread.currentThread().sleep(1000);
     } catch (InterruptedException e) {}
   }
 }
```

```
// all applets are graphical applications
  // the graphical appearance of the applet must be defined.
  // one way of doing so is to use the paint() method to paint the
appearance of the applet directly

  public void paint(Graphics g) {
    Calendar calendar = Calendar.getInstance();
    int hour = calendar.get(Calendar.HOUR_OF_DAY);
    int minute = calendar.get(Calendar.MINUTE);
    int second = calendar.get(Calendar.SECOND);
    g.setFont(font);
    g.setColor(color);
    g.drawString(hour + ":" + minute / 10 + minute % 10 +
                    ":" + second / 10 + second % 10, 10, 60);
  }

}
```

```
<!--DigitalClockDemo.html-->
<HTML>
  <HEAD>
     <TITLE> Digital Clock Applet </TITLE>
  </HEAD>
<BODY BGCOLOR=white>
  <CENTER>
    <H1> The Digital Clock Applet</H1>
      <P>
        <APPLET CODE=DigitalClock.class
               WIDTH=250 HEIGHT=80>
        </APPLET>
  </CENTER>
  <p> <hr>
  <a href = DigitalClock.java> The source </a>
</BODY>
</HTML>
```

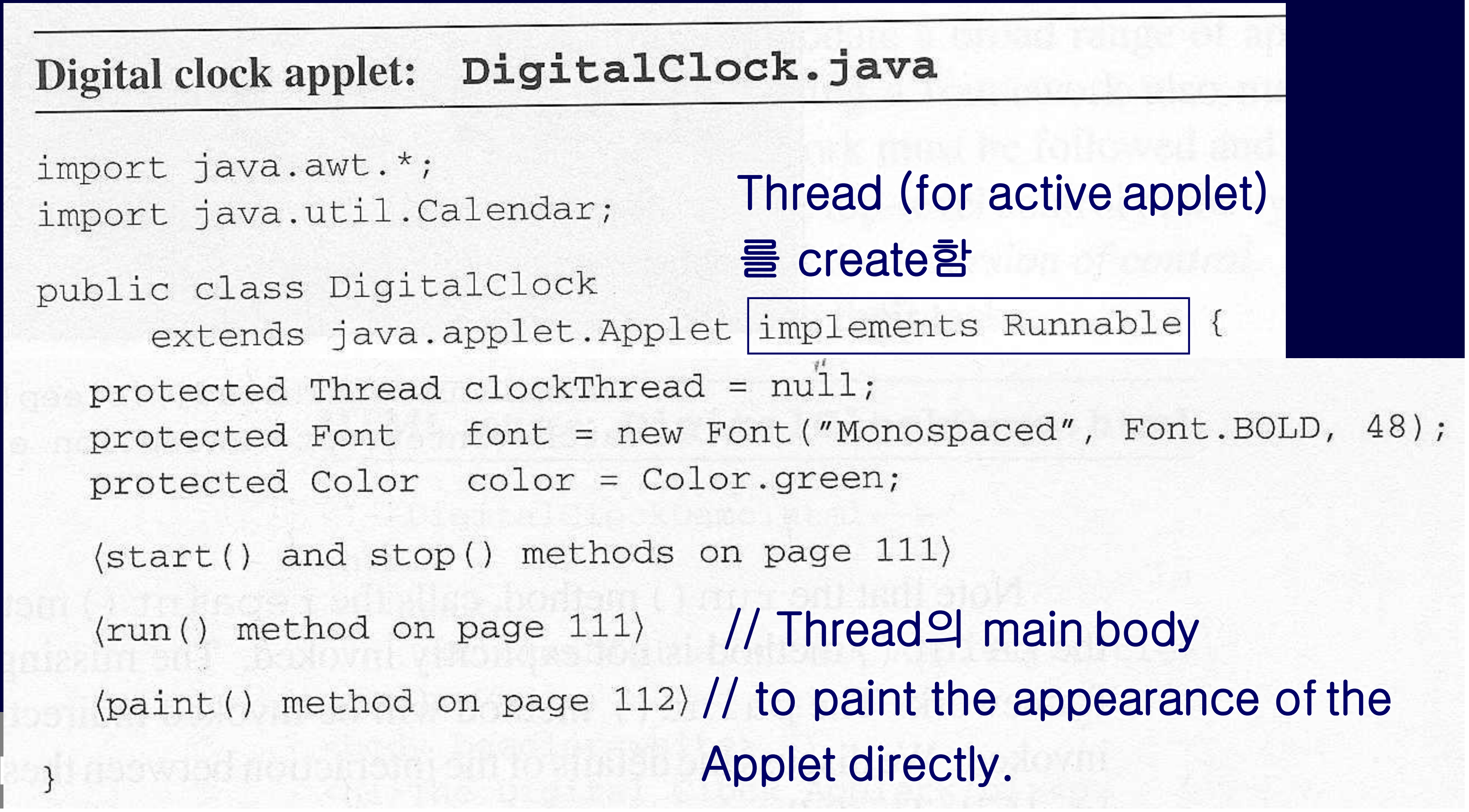


12:05:20

애플릿을 시작하였습니다.

# Ex. 4.1. A Digital Clock Applet – Initial Version

- Overall structure of the program.



**Digital clock applet:** `DigitalClock.java`

```
import java.awt.*;
import java.util.Calendar;

public class DigitalClock
    extends java.applet.Applet implements Runnable {
  protected Thread clockThread = null;
  protected Font   font = new Font("Monospaced", Font.BOLD, 48);
  protected Color  color = Color.green;

  ⟨start() and stop() methods on page 111⟩

  ⟨run() method on page 111⟩

  ⟨paint() method on page 112⟩
}
```

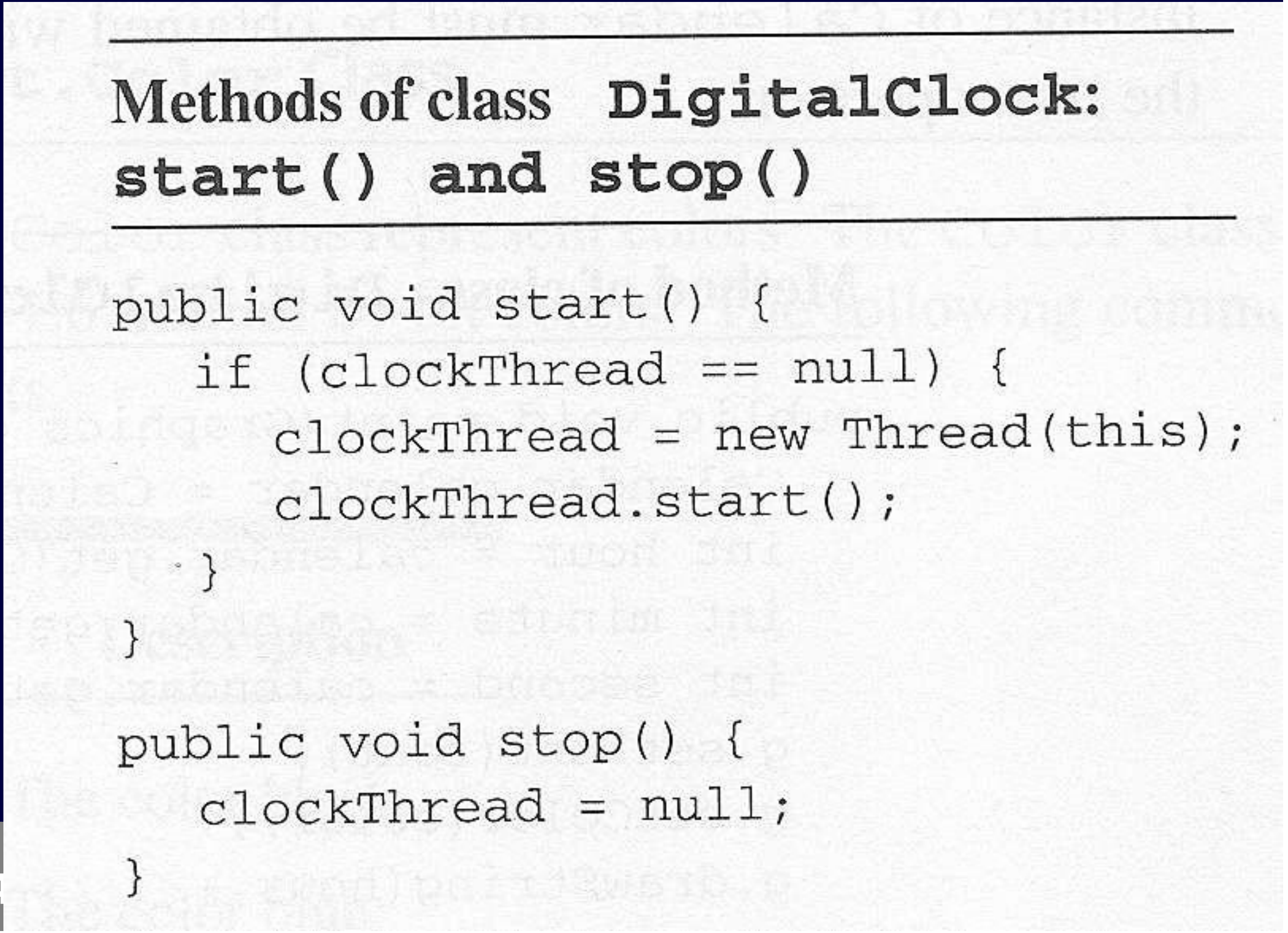
Thread (for active applet)를 create함

// Thread의 main body

// to paint the appearance of the Applet directly.

# Ex. 4.1. A Digital Clock Applet – Initial Version

- Start() and Stop()
  - Activate and deactivat e th e applet by creati ng and ki lling the t hread

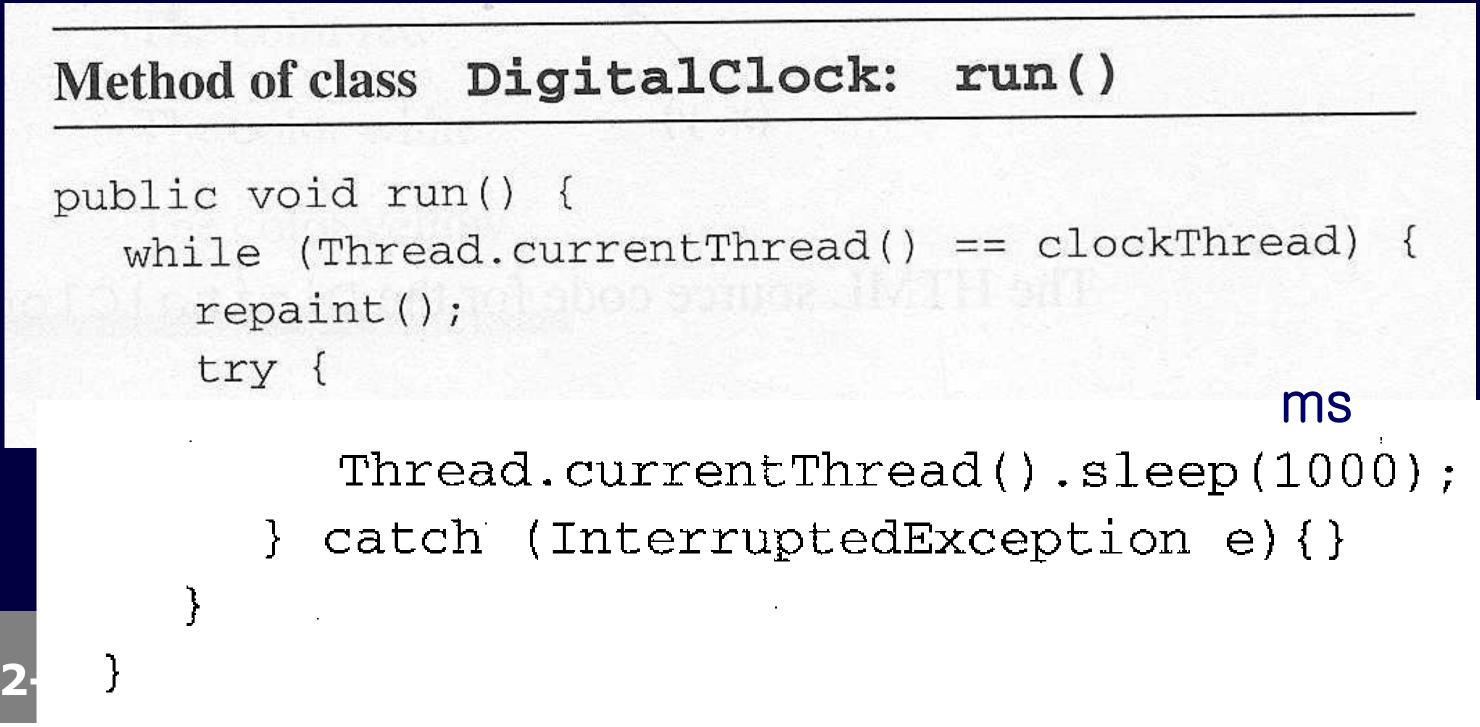


**Methods of class `DigitalClock`: `start()` and `stop()`**

```
public void start() {
   if (clockThread == null) {
      clockThread = new Thread(this);
      clockThread.start();
   }
}

public void stop() {
   clockThread = null;
}
```

# Ex. 4.1. A Digital Clock Applet – Initial Version

- Run()
  - -- infinite loop that periodically invokes the repaint()
  - refresh rate --- sleep의 argument에 의해 결정



**Method of class DigitalClock: run()**

```
public void run() {
   while (Thread.currentThread() == clockThread) {
      repaint();
      try {
         Thread.currentThread().sleep(1000);
      } catch (InterruptedException e){}
   }
}
```

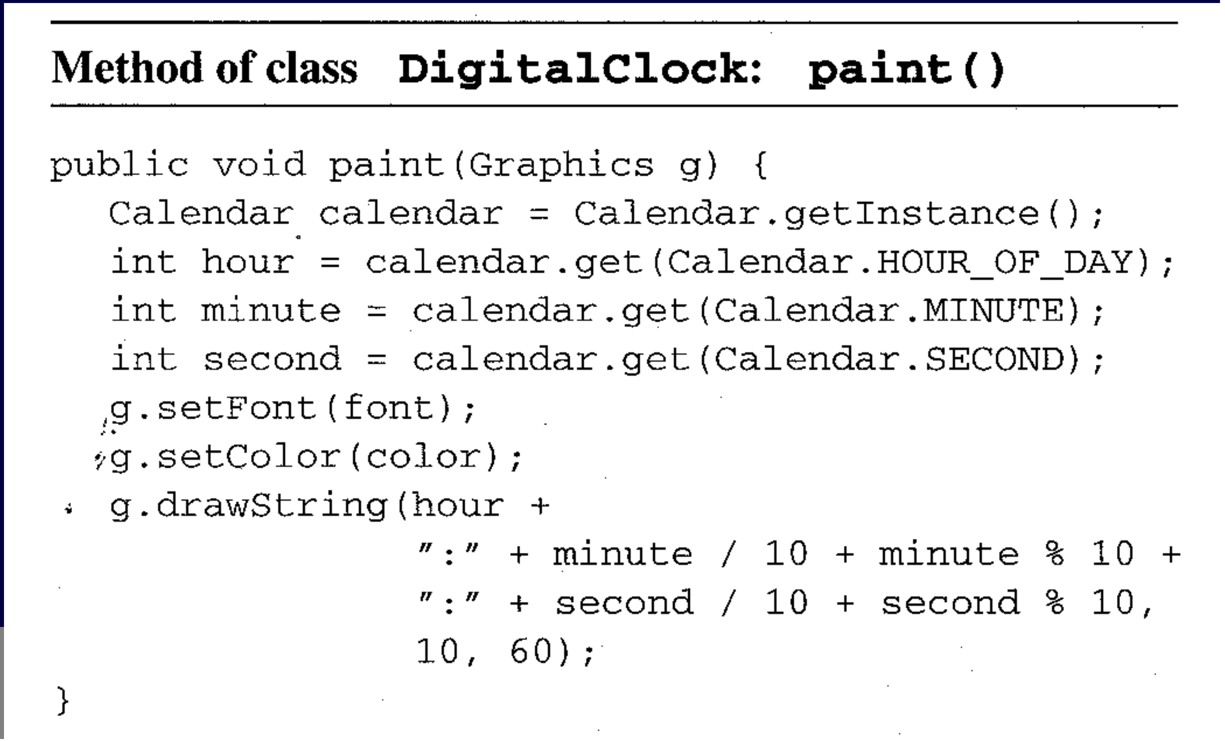
ms

## Ex. 4.1. A Digital Clock Applet – Initial Version

- Sleep methods
  - May throw an InterruptedException
  - Must be invoked inside a try-catch statements.
- Run() methods
  - Missing link – repaint(), paint()
  - the paint() method will be invoked indirectly when repaint() is invoked.
  - call the repaint() method, not paint(), to change the applet's appearance (provided by framework)
    - Sec 5.5
  - Override the paint(), not repaint() , to describe how the applet should be drawn.

# Ex. 4.1. A Digital Clock Applet – Initial Version

- calendar – singleton class
  - Instance of Calendar must be obtainedwith the getInstance(), not the new operator.

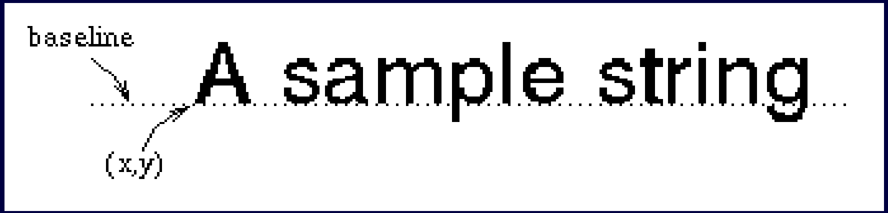


**Method of class DigitalClock: paint()**

```
public void paint(Graphics g) {
   Calendar calendar = Calendar.getInstance();
   int hour = calendar.get(Calendar.HOUR_OF_DAY);
   int minute = calendar.get(Calendar.MINUTE);
   int second = calendar.get(Calendar.SECOND);
   g.setFont(font);
   g.setColor(color);
   g.drawString(hour +
               ":" + minute / 10 + minute % 10 +
               ":" + second / 10 + second % 10,
               10, 60);
}
```

## Ex. 4.1. A Digital Clock Applet – Initial Version

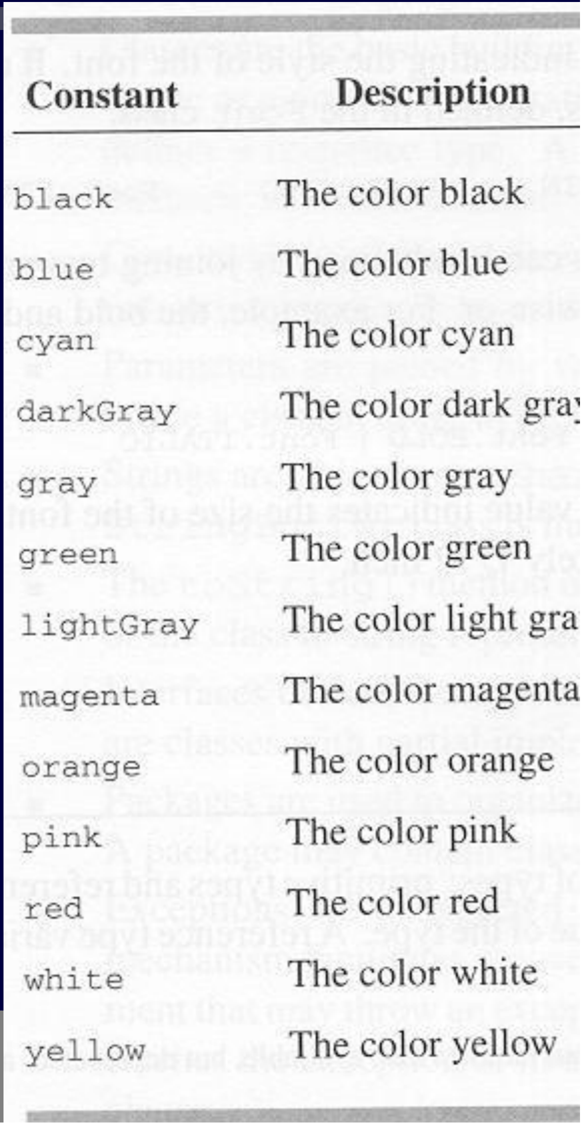
- drawString()
  - three argument drawString(str,x,y)
  - x,y : left end of the string on the baseline



## HTML source: DigitalClockDemo.html

```
<!--DigitalClockDemo.html-->
<html>
   <head>
      <title>Digital Clock Applet</title>
   </head>
<body bgcolor=white>
<h1>The Digital Clock Applet</h1><p>
<applet code=DigitalClock.class
        width=250 height=80>
</applet>
<p><hr>
<a href=DigitalClock.java>The source</a>
</body>
</html>
```

- The java.awt.color Class
  - Color Class
    - 1.6 million , 24bit colors
    - Create an color
      - new Color(r,g,b)
      - r,g,b : range 0 to 255
- The java.awt.Font Class
  - new Font(name,style,size)



| Constant | Description |
|---|---|
| black | The color black |
| blue | The color blue |
| cyan | The color cyan |
| darkGray | The color dark gray |
| gray | The color gray |
| green | The color green |
| lightGray | The color light gra |
| magenta | The color magenta |
| orange | The color orange |
| pink | The color pink |
| red | The color red |
| white | The color white |
| yellow | The color yellow |