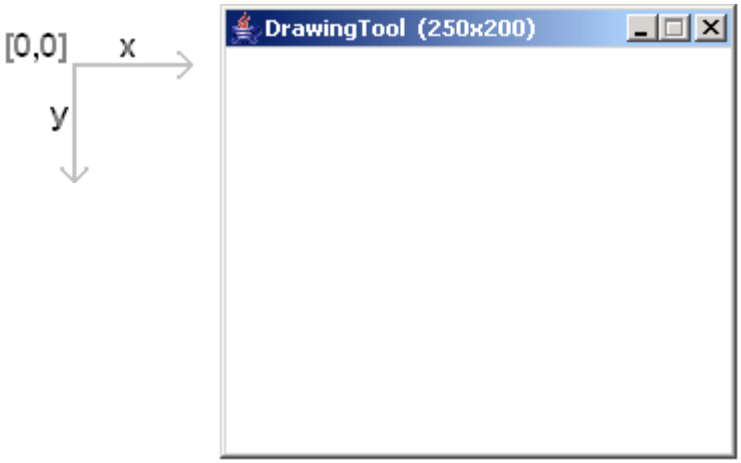
Třída DrawingTool

1. Základ

Třída DrawingTool je určena k jednoduchému kreslení pomocí několika základních příkazů do grafického okna zadaných rozměrů (nastavení v konstruktoru), které se při vytvoření instance této třídy objeví na obrazovce (viz Obrázek 1).

```
public static void main(String[] args) {
      // DrawingTool dt = new DrawingTool(int width, int height);
      DrawingTool dt = new DrawingTool(250,200);
}
```



Obrázek 1: Prázdné okno připravené pro kreslení

Důležité je si uvědomit, že souřadný systém má svůj počátek v bodě [0,0]. Tento bod se nalézá v levém horním rohu grafického okna, přičemž směrem doprava roste souřadnice **x** a směrem dolů roste souřadnice **y** (viz Obrázek 1).

Velikost grafického okna (šířka, výška) je zadávána v pixelech (obrazovkových bodech). Je tedy vhodné nenastavovat velikost okna větší než je aktuální nastavené rozlišení pracovní plochy vašeho operačního systému (běžně bývá např. 1024x768).

Základními příkazy, které třída DrawingTool poskytuje, jsou

- `line(int x1, int y1, int x2, int y2);`
  - Vykreslí čáru od bodu (x1,y1) do bodu (x2,y2) včetně,

a

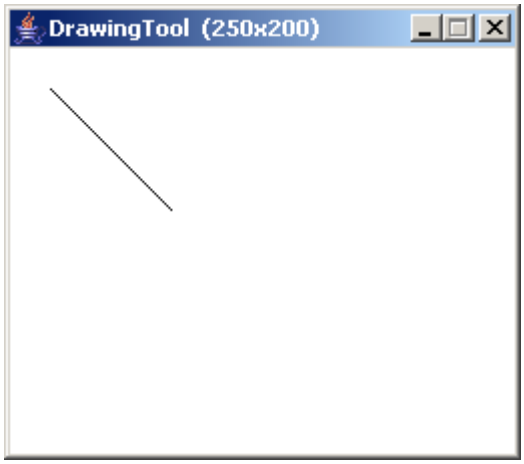
- `clear();`
  - Smaže obsah celého okna.

Pokud tedy budeme chtít vykreslit čáru, která bude začínat v bodu [20,20] a končit v bodu [80,80], bude programový kód vypadat následovně:

```
public static void main(String[] args) {
      int width = 250;
      int height = 200;

      DrawingTool dt = new DrawingTool(width, height);
      dt.line(20, 20, 80, 80);
}
```

a výstup programu bude vypadat takto:



Obrázek 2: Vykreslená čára z bodu [20,20] do bodu [80,80]

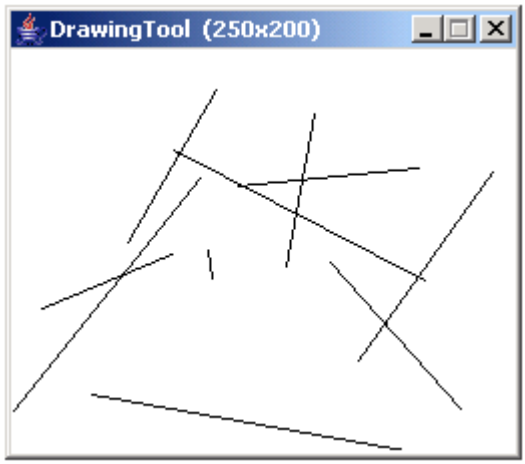
Chceme-li do okna náhodně vykreslit 10 čar, bude programový kód vypadat například následovně:

```
public static void main(String[] args) {
      int width = 250;
      int height = 200;

      DrawingTool dt = new DrawingTool(width, height);

      for (int i=0; i<10; i++) {
            int x1 = (int) (Math.random()*width)  + 1;
            int y1 = (int) (Math.random()*height) + 1;
            int x2 = (int) (Math.random()*width)  + 1;
            int y2 = (int) (Math.random()*height) + 1;
            dt.line(x1,y1,x2,y2);
      }
}
```

Výstup tohoto programu viz Obrázek 3.



Obrázek 3: Výstup předchozího programového kódu – náhodné čáry

Obdobně lze voláním metody `dt.clear()` obrazovku smazat a začít znovu kreslit do prázdného okna.

2. Další možnosti

2.1 Nastavení barev

Barva pozadí okna, do kterého se kreslí, je implicitně bílá a barva kreslených čar je implicitně černá (viz Obrázek 3). Barvu pozadí je možné definovat v konstruktoru

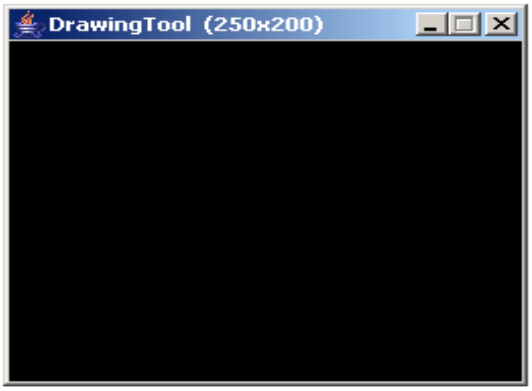
```
DrawingTool dt = new DrawingTool(int width, int height, Color color);
```

Aby bylo možné definovat barvy, je **vždy** nutné importovat balík *java.awt.Color;* (popis viz Java dokumentace). Pokud bychom tedy chtěli nastavit barvu pozadí okna na černou, bude programový kód vypadat následovně:

```
import java.awt.Color;
...
   public static void main(String[] args) {
        int width = 250;
        int height = 200;

        DrawingTool dt = new DrawingTool(width, height, Color.BLACK);
   }
```

Výstup tohoto programu viz Obrázek 4.



Obrázek 4: Výstup předchozího programového kódu

Pokud bychom chtěli změnit barvu vykreslovaných čar, je k dispozici metoda `setColor()` s parametrem barvy popředí (tedy barvy vykreslovaných objektů):

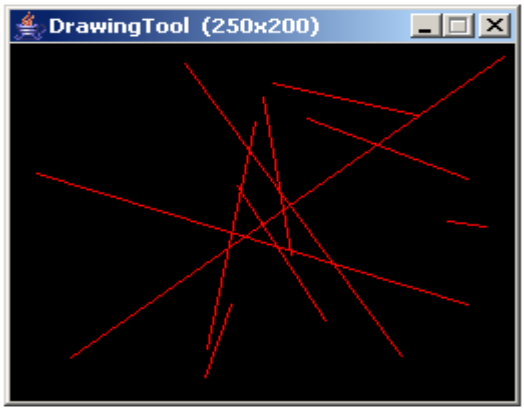
```
setColor(Color fgColor);
```

Ukázka program. kódu a výstupu s definováním vlastních barev pozadí a vykreslovaných čar:

```
public static void main(String[] args) {
      int width = 250;
      int height = 200;

      DrawingTool dt = new DrawingTool(width, height, Color.BLACK);
      dt.setColor(Color.RED);
      for (int i=0; i<10; i++) {
            int x1 = (int) (Math.random()*width)  + 1;
            int y1 = (int) (Math.random()*height) + 1;
            int x2 = (int) (Math.random()*width)  + 1;
            int y2 = (int) (Math.random()*height) + 1;
            dt.line(x1, y1, x2, y2);
      }
}
```

Výstup tohoto programu viz Obrázek 5.



Obrázek 5: Výstup předchozího programu – vlastní barvy

Poznámka:
Při definování vlastních barev je potřeba být obezřetný. Může se stát, že nedopatřením nastavíme stejnou barvu pozadí okna i barvy vykreslovaných čar. Při spuštění se potom může zdát, že váš program nic nedělá, že nefunguje. Je zbytečné nad takovouto záležitostí trávit dlouhé chvíle a hledat chybu ve vašem programu – žádná tam vlastně není.

2.2 Antialiasing

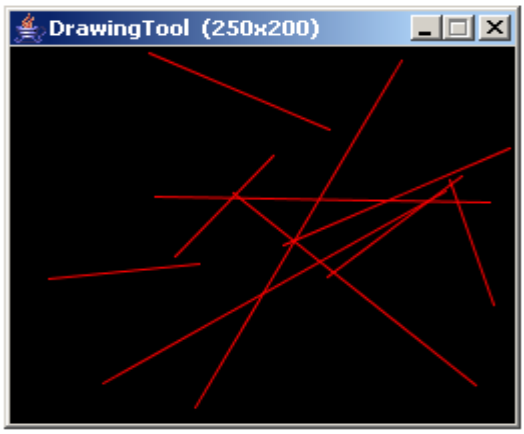
Při kreslení čar je možné zapnout antialiasing[1], čímž bude vaše vykreslená čára vizuálně hladší (potlačí se chyby v diskretizaci). Zapnout antialiasing je možné v konstruktoru

```
DrawingTool dt = new DrawingTool(int width, int height, Color color, boolean antialiasing);
```

nebo v konstruktoru

```
DrawingTool dt = new DrawingTool(int width, int height, boolean antialiasing);
```

Na obrázku 6 je zobrazen výstup programu z předchozí kapitoly se zapnutým antialiasingem.



Obrázek 6: Výstup programu z předchozí kapitoly se zapnutým antialiasingem

Poznámka:
Pokud vám záleží na rychlosti běhu vašeho programu, antialiasing nezapínejte, protože zpomaluje vykreslování, což je patrné především na slabších počítačích.

2.3 Animace

Třída DrawingTool je koncipována tak, aby její použití bylo co nejjednodušší. Protože obsah okna je potřeba uchovávat, aby při překrytí jiným oknem nedošlo ke ztrátě obsahu, je v mechanismu vykreslování implementován princip double bufferingu[2]. Při požadavku na

[1] http://en.wikipedia.org/wiki/Antialiasing
[2] http://en.wikipedia.org/wiki/Double_buffering

vykreslení je grafický objekt (například čára) ihned zapsán do paměťového bufferu (back bufferu) a ten je potom okamžitě zkopírován do obrazovkového bufferu (screen bufferu). To je samozřejmě pro účely animace nevhodné, protože toto okamžité zobrazení na obrazovce po provedení jakékoliv změny v obraze způsobuje nepříjemné blikání obrazu (flickering), nemluvě o zbytečném zpomalení vykreslování vlivem kopírování obsahu back bufferu do screen bufferu.

Tuto skutečnost je možné změnit pomocí metod

- `stopUpdate();`
  - zakáže vykreslování grafických objektů na obrazovku – kreslení bude realizováno pouze do paměťového bufferu (back bufferu)

a

- `startUpdate();`
  - opět povolí vykreslování grafických objektů na obrazovku – opět bude docházet ke kopírování paměťového bufferu (back bufferu) do obrazovkového bufferu (screen bufferu)

Ukázka použití metod `stopUpdate()` a `startUpdate()` při realizaci animace:

```
import java.awt.Color;
.....
public static void main(String[] args) {
	int count = 200;
	int width = 600;
	int height = 600;
	int[][] body = new int[count*2][2];
	int[][] smer = new int[count*2][2];
	int[] barva = new int[count];
	Color[] barvy = {
			Color.BLACK, Color.BLUE, Color.CYAN,
			Color.DARK_GRAY, Color.GRAY, Color.GREEN,
			Color.LIGHT_GRAY, Color.MAGENTA,
			Color.ORANGE, Color.PINK, Color.RED,
			Color.WHITE, Color.YELLOW
	};
	DrawingTool dt = new DrawingTool(width, height, Color.BLACK, false);
	for (int i=0; i<count; i=i+2) {
		body[i][0] = (int) (Math.random()*width) + 1;
		body[i][1] = (int) (Math.random()*height) + 1;
		body[i+1][0] = (int) (Math.random()*width) + 1;
		body[i+1][1] = (int) (Math.random()*height) + 1;
		smer[i][0] = (Math.random()*(width/2)) > width/2 ? 1 : -1;
		smer[i][1] = (Math.random()*height) > height/2 ? 1 : -1;
		smer[i+1][0] = (Math.random()*width) > height/2 ? 1 : -1;
		smer[i+1][1] = (Math.random()*height) > height/2 ? 1 : -1;
		barva[i] = (int)(Math.random()*barvy.length);
		dt.setColor(barvy[barva[i]]);
		dt.line(body[i][0], body[i][1], body[i+1][0], body[i+1][1]);
	}

	do {
		dt.stopUpdate();
		dt.clear();
		for (int i=0; i<count; i=i+2) {
			body[i][0] += smer[i][0];
			body[i][1] += smer[i][1];
			body[i+1][0] += smer[i+1][0];
			body[i+1][1] += smer[i+1][1];
			if (body[i][0]>=width || body[i][0]<=0)
				smer[i][0] = -smer[i][0];
			if (body[i][1]>=height || body[i][1]<=0)
				smer[i][1] = -smer[i][1];
			if (body[i+1][0]>=width || body[i+1][0]<=0)
				smer[i+1][0] = -smer[i+1][0];
			if (body[i+1][1]>=height || body[i+1][1]<=0)
				smer[i+1][1] = -smer[i+1][1];
			dt.setColor(barvy[barva[i]]);
			dt.line(body[i][0], body[i][1], body[i+1][0], body[i+1][1]);
		}
		dt.startUpdate();
	} while (1==1);
}
```

Poznámka:

Pokud zakomentujete příkazy `dt.stopUpdate()` a `dt.startUpdate()`, *uvidíte při spuštění programu efekt blikání a pomalého vykreslování, které použití těchto příkazů odstraňuje. Obecný cyklus jak dělat animace je:*

1. *zakázat vykreslování na obrazovku příkazem* `dt.stopUpdate();`
2. *připravit scénu (klasicky vykreslit objekty – kreslení bude probíhat do paměťového bufferu (back bufferu);*
   (v předchozím kódu tento bod odpovídá smazání obsahu okna, posunu bodů jednotlivých čar na novou pozici a následně jejich vykreslení na nových pozicích)
3. *povolit vykreslování na obrazovku příkazem* `dt.startUpdate()` *(ihned dojde k vykreslení paměťového bufferu na obrazovku)*