

Interfaces graphiques et programmation évènementielle

Célia Martinie – martinie@irit.fr

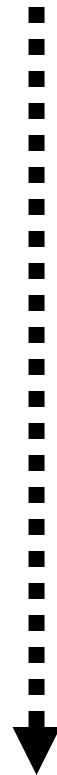
Mathieu Raynal – raynal@irit.fr

Chargés de TP: Christophe Bortolaso, Célia Martinie, Mathieu Raynal, Philippe Truillet



Exécution typique d'un programme

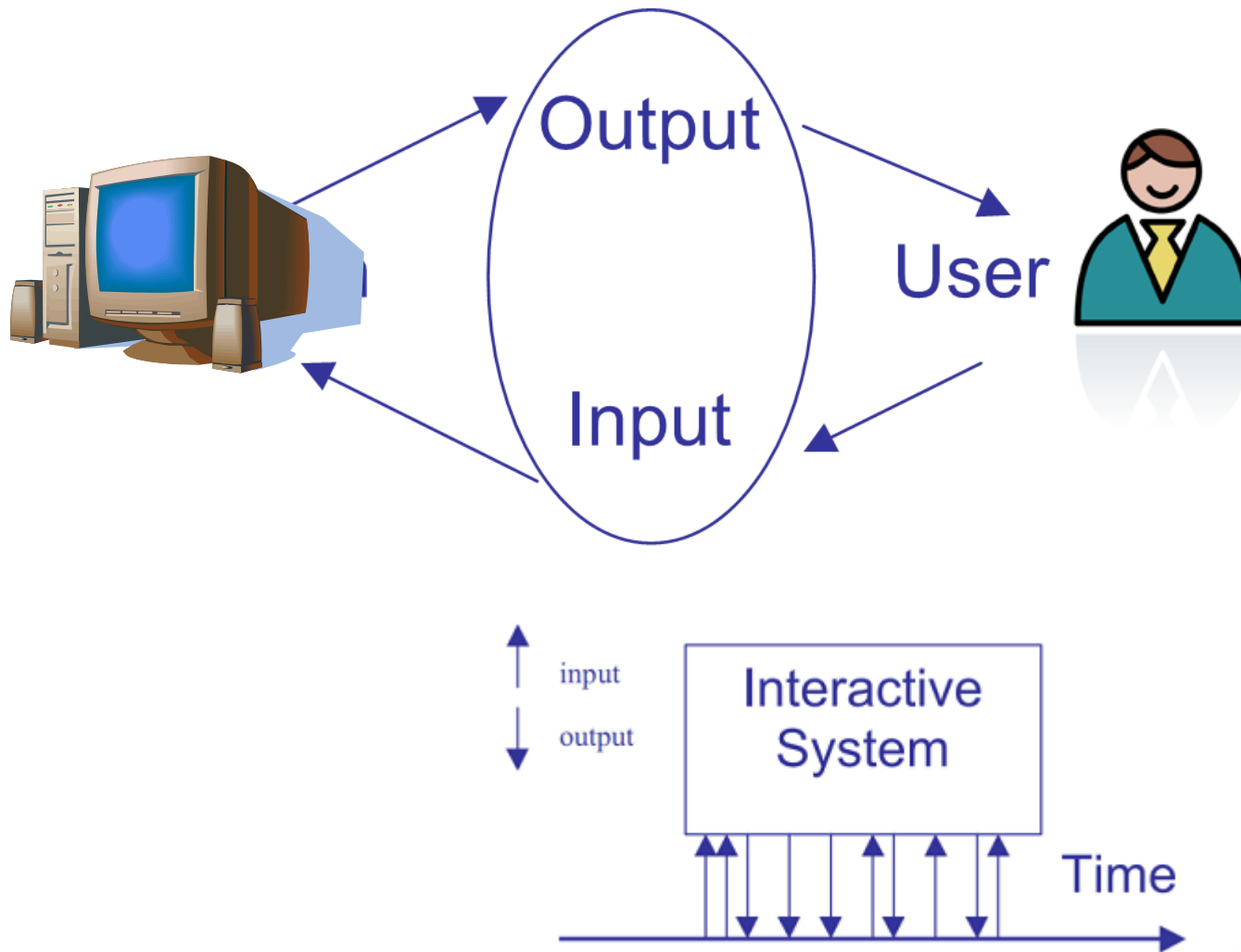
- Non-interactive
- Exécution linéaire
- Processus d'automatisation (automatique)
- Ne prend pas en compte les capacités de l'humain versus ordinateur



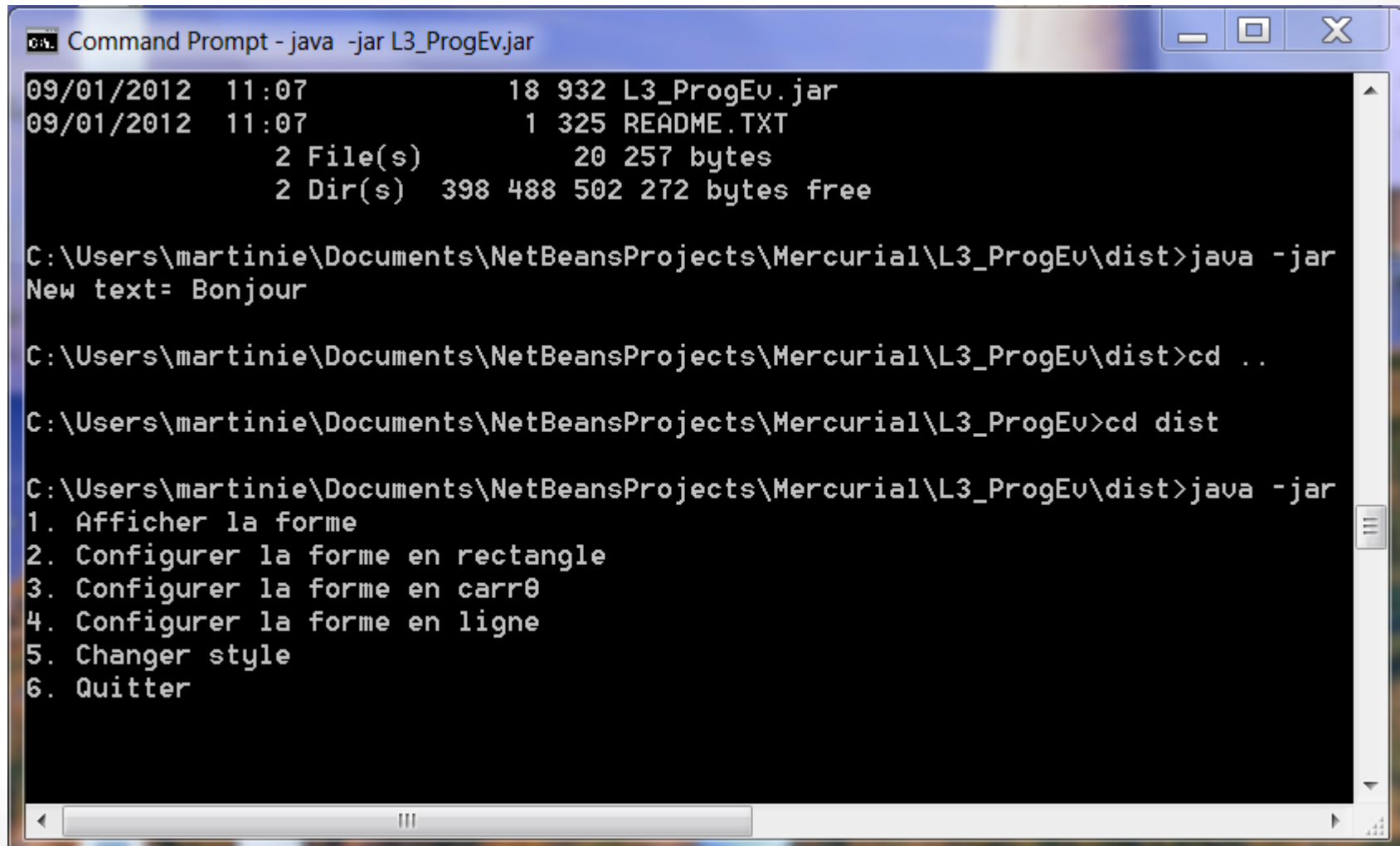
program:

```
main()
{
    code;
    code;
    code;
    code;
    code;
    code;
    code;
    code;
    code;
    code;
    code;
    code;
}
```

Systeme interactif



Ligne de commande



```
09/01/2012 11:07      18 932 L3_ProgEv.jar
09/01/2012 11:07       1 325 README.TXT
      2 File(s)      20 257 bytes
      2 Dir(s)  398 488 502 272 bytes free

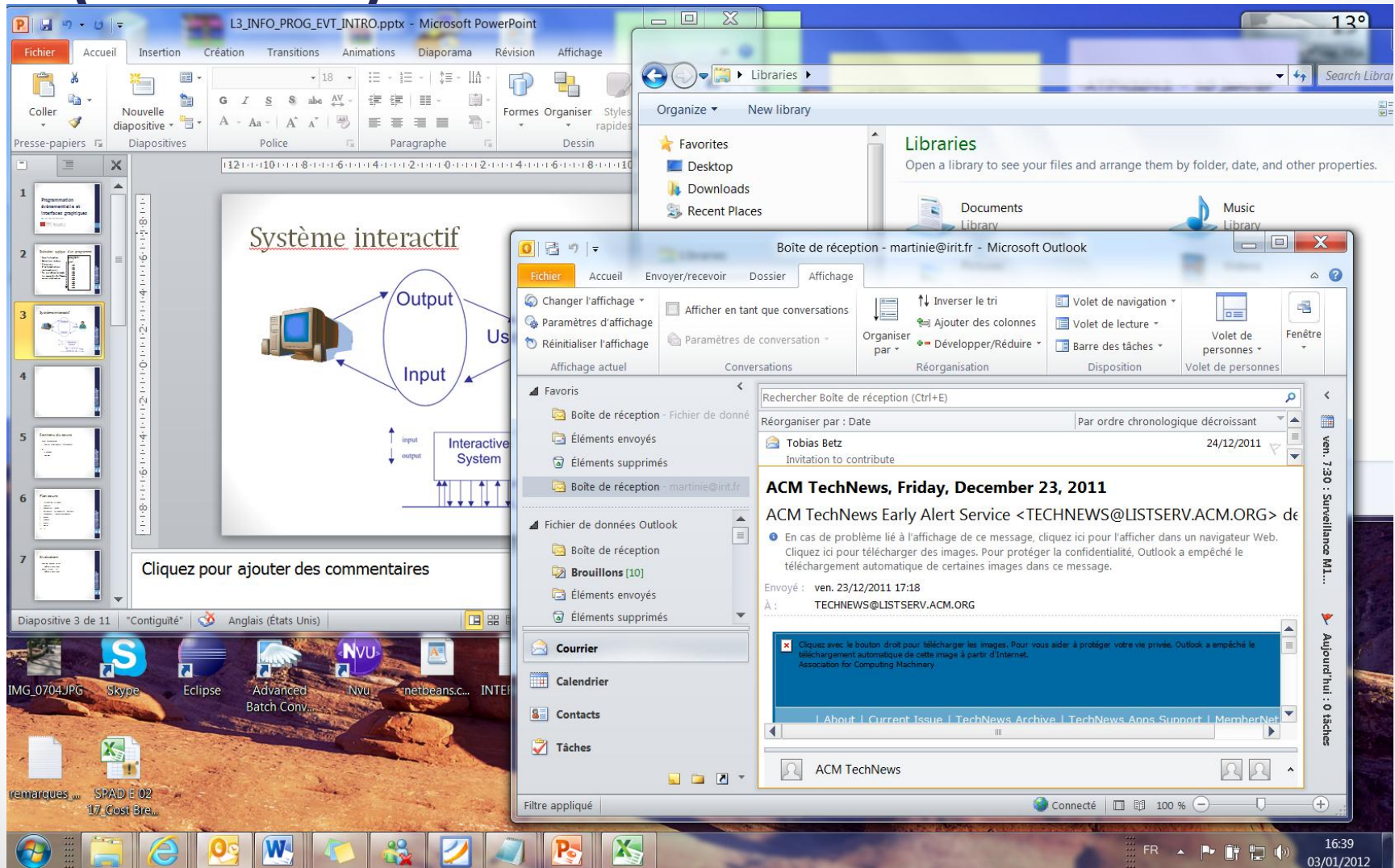
C:\Users\martinie\Documents\NetBeansProjects\Mercurial\L3_ProgEv\dist>java -jar
New text= Bonjour

C:\Users\martinie\Documents\NetBeansProjects\Mercurial\L3_ProgEv\dist>cd ..

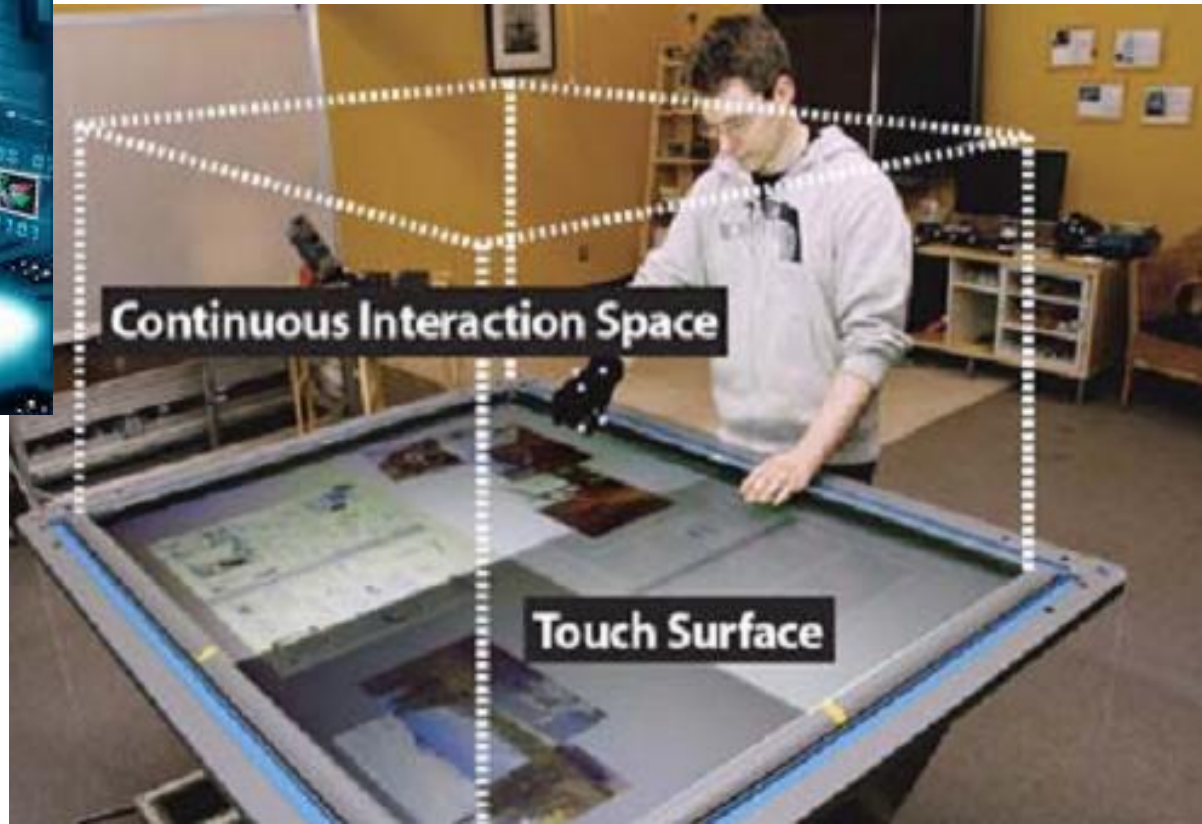
C:\Users\martinie\Documents\NetBeansProjects\Mercurial\L3_ProgEv>cd dist

C:\Users\martinie\Documents\NetBeansProjects\Mercurial\L3_ProgEv\dist>java -jar
1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
```

Window Icon Menu Pointer (WIMP)



Interfaces post-WIMP



Interfaces utilisateur

- Ligne de commande (ou command-line)
- Interfaces graphiques (ou GUI)
- Ne font pas l'objet de ce cours
 - Commande/synthèse vocale
 - Détection de mouvement

Contenu du cours

- Conception d'interfaces par composants
- Programmation par événement
- Bibliothèque Java Swing
- Plateforme de développement NetBeans

Plan du cours

1. Introduction et rappels
2. Fenêtres
3. Composants simples
4. Gestionnaire de géométrie, Containers
5. Evènements (sources, gestionnaire)
6. Menus
7. Tableaux
8. Arbres
9. Dessin

Evaluation

- Contrôle Continu en TP
 - 30% de la note finale
- **Contrôle terminal sur table (2h)**
 - 70% de la note finale

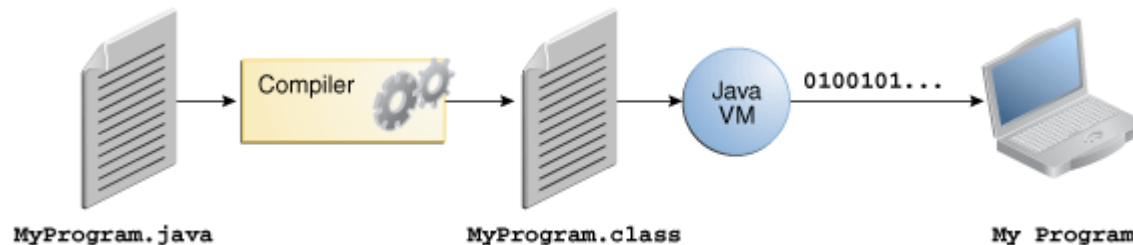
COURS-TD 1: INTRODUCTION ET RAPPELS

Rappels JAVA

- Langage de programmation libre
 - Développé en 1995 par Sun puis racheté par Oracle (2009)
- Portable « write once, run everywhere »
- Interprété (Bytecode et JVM)
- Orienté Objet
- Une bibliothèque de classes (API)
- Liste de rappels non exhaustive, bases nécessaires pour le cours

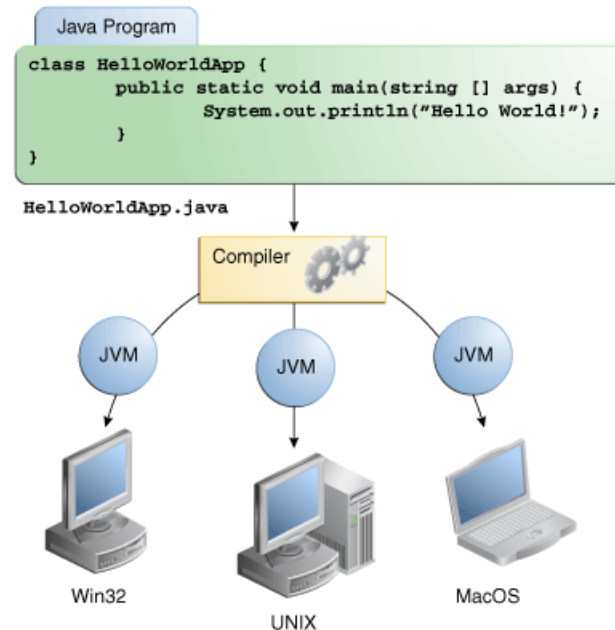
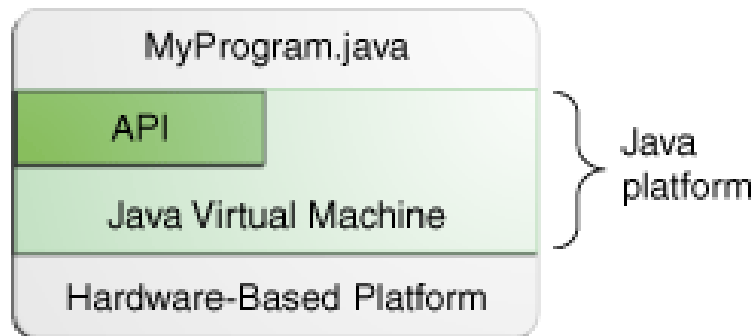
Java Virtual Machine (JVM)

- Machine virtuelle
 - le compilateur Java génère du byte code et non de l'assembleur
 - le byte code est exécuté sur une machine virtuelle : la JVM



Avantages

- Programmation de plus haut niveau (plus de gestion manuelle de la mémoire)
- Indépendance de la plateforme cible : windows, linux, mac, web, smartphone, ...



Exercice 1 - HelloWorld

Ecrire un programme java permettant d'afficher *HelloWorld* en ligne de commande

Exercice 1 - HelloWorld

```
/**
```

```
 * The HelloWorldApp class implements an application that
```

```
 * simply displays "Hello World!" to the standard output.
```

```
 */
```

```
class HelloWorldApp {
```

```
    public static void main(String[] args) {
```

```
        System.out.println("Hello World!"); //Display the string.
```

```
    }
```

```
}
```

Bases

- Variables
- Opérateurs
- Expressions

Variables

- 3 types
 - Types primitifs
 - Références à un objet
 - Références à un tableau
- Utilisation
 - Instance
 - Classe (**static** – 1 seule variable qqsoit le nombre d'instances)
 - Locale
 - Paramètre
 - Constante (**final**)
- Types primitifs
 - short (entier 16 bits)
 - int (entier 32 bits)
 - long (entier 64 bits)
 - char (caractère unicode 16 bits)
 - float (non entier 32 bits)
 - boolean (false, true)
 - ...

DECLARATION et INITIALISATION

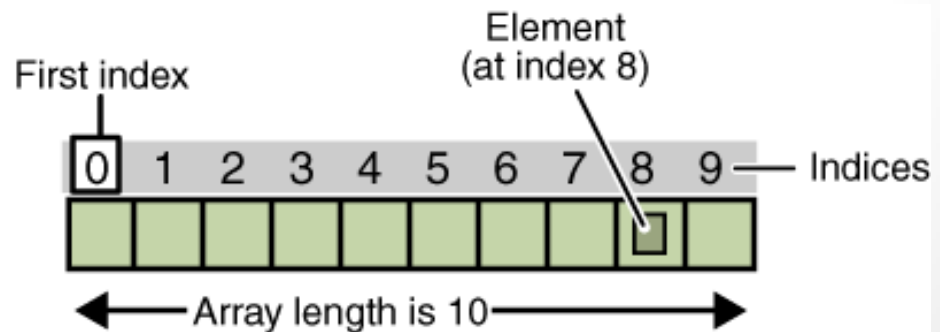
Variables - Tableaux

Exemple 1

```
// declares an array of  
integers  
int[] anArray;  
  
// create an array of integers  
anArray = new int[10];  
  
// initialize first element  
anArray[0] = 100;  
  
// initialize second element  
anArray[1] = 200;  
  
// etc. anArray[2] = 300;
```

Exemple 2

```
// declares an array of  
integers  
int[] anArray = { 100, 200,  
300, 400, 500, 600, 700, 800,  
900, 1000 };
```



Conventions de nommage

- **Variable names** are case-sensitive. A variable's name can be any legal identifier — an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "\$", or the underscore character "_". The convention, however, is **to always begin your variable names with a letter**, not "\$" or "_". Additionally, the dollar sign character, by convention, is never used at all. You may find some situations where auto-generated names will contain the dollar sign, but your variable names should always avoid using it. A similar convention exists for the underscore character; **while it's technically legal to begin your variable's name with "_", this practice is discouraged. White space is not permitted.**
- Subsequent characters may be letters, digits, dollar signs, or underscore characters. Conventions (and common sense) apply to this rule as well. When choosing a name for your variables, use full words instead of cryptic abbreviations. Doing so will make your code easier to read and understand. In many cases it will also make your code self-documenting; fields named cadence, speed, and gear, for example, are much more intuitive than abbreviated versions, such as s, c, and g. **Also keep in mind that the name you choose must not be a keyword or reserved word.**
- If the name you choose consists of only one word, **spell that word in all lowercase letters. If it consists of more than one word, capitalize the first letter of each subsequent word.** The names gearRatio and currentGear are prime examples of this convention. If your variable stores a **constant value, such as static final int NUM_GEAR = 6, the convention changes slightly, capitalizing every letter and separating subsequent words with the underscore character.** By convention, the underscore character is never used elsewhere.

Conventions

- Une seule classe par fichier. Même nom que la classe
- Noms des classes et interfaces commencent par une majuscule
- Noms des attributs commencent par une minuscule
- Noms des packages en minuscule
- Constantes en majuscule
- Tout doit être dans une classe

Opérateurs

- Affectation =
- Incrémentation/Décrémentation ++, --
- Arithmétiques +, -, *, /
- Egalité et comparaison ==, !=, >, <, >=, <=
- Logiques !, &, ||

```
int i = 4;
```

```
i++;
```

```
i = i + 2;
```

```
i--;
```

Exercice 2: que renverrait $i \geq 6$?

Expressions - if

if

```
boolean count = true;  
int countVal = 0;  
  
...  
if (count){  
    countVal++;  
}
```

if - else

```
boolean count = false;  
int countVal = 0;  
  
...  
if (count){  
    countVal++;  
}  
else{  
    System.out.println("Counter  
stopped");  
}
```

Expressions - switch

```
int month = 8;
String monthString;
switch (month) {
    case 1:
        monthString = "January";
        break;
    case 2:
        monthString = "February";
        break;
    case 3:
        monthString = "March";
        break;
    case 4:
        monthString = "April";
        break;
    case 5:
        monthString = "May";
        break;
    case 6:
        monthString = "June";
        break;
    case 7:
        monthString = "July";
        break;
    case 8:
        monthString = "August";
        break;
    case 9:
        monthString = "September";
        break;
    case 10:
        monthString = "October";
        break;
    case 11:
        monthString = "November";
        break;
    case 12:
        monthString = "December";
        break;
    default:
        monthString = "Invalid
month";
        break;
}
```

Exercice 3: valeur de monthString en sortie?

Expressions – while et do-while

```
class WhileDemo {  
    public static void main(String[] args){  
        int count = 1;  
        while (count < 11) {  
            System.out.println("Count is: " + count);  
            count++;  
        }  
    }  
}
```

Exercice 4: écrire les sorties des programmes

```
class DoWhileDemo {  
    public static void main(String[] args)  
    {  
        int count = 1;  
        do {  
            System.out.println("Count is: " + count);  
            count++;  
        } while (count <= 11);  
    }  
}
```

Expressions - for

```
class ForDemo {  
    public static void main(String[] args){  
        for(int i=1; i<11; i++){  
            System.out.println("Count is: " + i);  
        }  
    }  
}
```

Exercice 5: écrire la sortie du programme

Rappels P00

- Objet
- Classe et instance de classe
- Interface
- Héritage

Objet

- Identité (hashcode)
- Etat
- Comportement

Instance d'une classe

Classe

- Attributs (private, public)
- Méthodes, fonctions
- Constructeur

```
public class TextArea {  
  
    private int width;  
    private int height;  
    private String text;  
  
    public TextArea () {  
        width = 50;  
        height = 10;  
        text = "";  
    }  
  
    public void setText(String newText) {  
        this.text = newText;  
    }  
  
    ...  
}
```


Instanciación

```
TextArea welcomeMessage = new TextArea ();
```

welcomeMessage

- est un objet
- contient une instance de la classe *TextArea*

Exercice:

```
public class MyApplication{  
  
    public static void main(String[] args) {  
        TextArea welcomeMessage;  
        welcomeMessage.setText("Bonjour");  
    }  
}
```

Exercice 6: Quel est le problème dans ce programme?

Interface

- Spécification de la classe à implémenter
- Partage des développements logiciels

```
public interface FontTheme {  
    public void setFontName (String name);  
    public void setFontSize (int size);  
    public void setFontColor (Color color);  
}
```

Interface - exemple

```
public class TextArea implements FontTheme{
```

```
    private int width;  
    private int height;  
    private String text;  
    private String fontName;
```

```
    public TextArea(){  
        width = 50;  
        height = 15;  
        text = "";  
    }  
  
    public void setText(String newText){  
        this.text = newText;  
        System.out.println("New text= "+newText);  
    }
```

```
        @Override  
        public void setFontName(String name)  
        {  
            fontName = name;  
        }  
        ...  
    }
```

Interface – exemple 2

```
public class TextArea implements FontTheme{
```

```
    private int width;
```

```
    private int height;
```

```
    private String text;
```

```
    private String fontName;
```

```
    private int fontSize;
```

```
    private Color fontColor;
```

```
    public TextArea(){
```

```
        width = 50;
```

```
        height = 15;
```

```
        text = "";
```

```
    }
```

```
    public void setText(String newText){
```

```
        this.text = newText;
```

```
        System.out.println("New text= "+newText);
```

```
    }
```

```
        @Override
```

```
        public void setFontName(String name)
        {
```

```
            fontName = name;
```

```
        }
```

```
        public void setFontSize(int size){
```

```
            font Size = size;
```

```
            ...
```

```
        }
```

```
        public void setFontColor (Color c){
```

```
            fontColor = c;
```

```
            ...
```

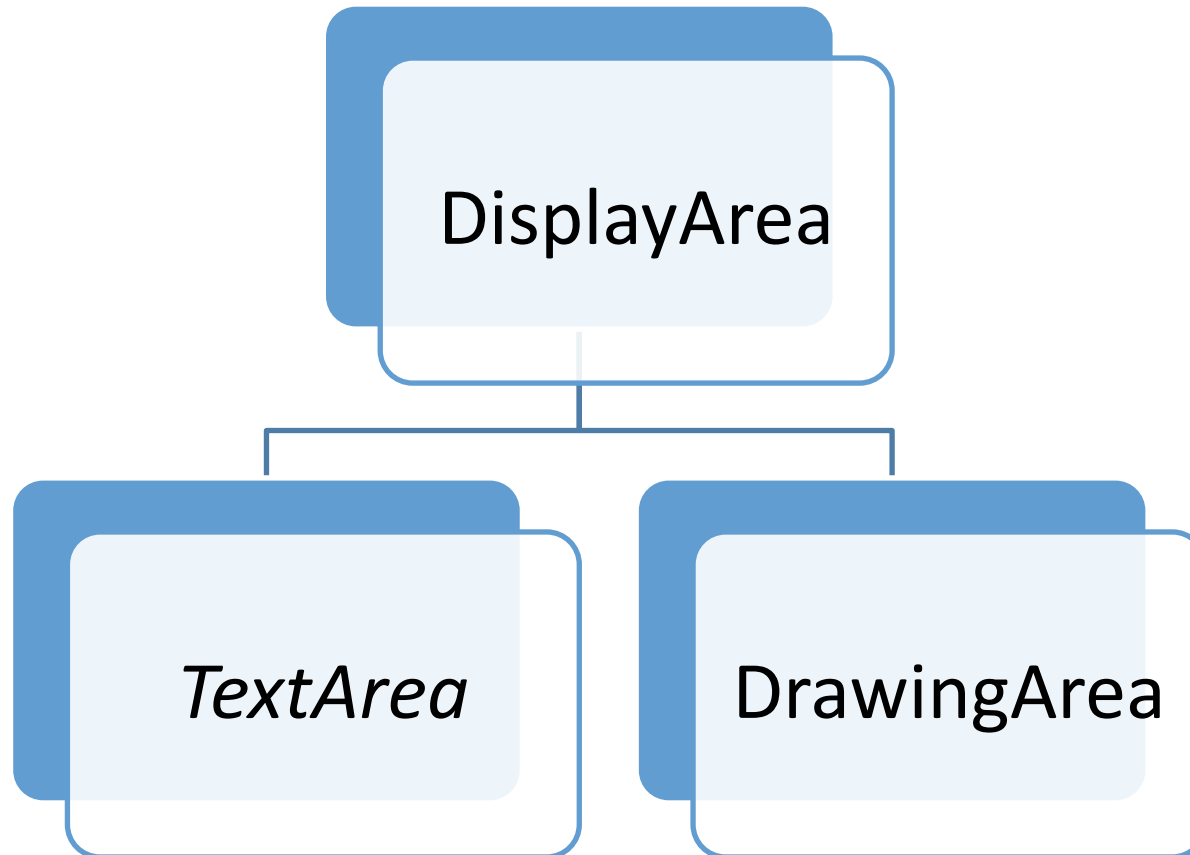
```
        }
```

```
        ...
```

```
    }
```

Héritage

- Hiérarchie de classes



DisplayArea

```
public class DisplayArea {  
    int width;  
    int height;  
  
    public DisplayArea(){  
        width = 50;  
        height = 15;  
    }  
  
    public int getWidth() {  
        return width;  
    }  
  
    public int getHeight() {  
        return height;  
    }  
}
```

TextArea

```
public class TextArea extends DisplayArea implements FontTheme{
```

```
    private String text;  
    private String fontName;
```

```
    public TextArea(){  
        text = "";  
    }
```

```
    public void setText(String newText){  
        this.text = newText;  
        System.out.println("New text= "+newText);  
    }
```

```
    @Override  
    public void setFontName(String name) {  
        this.fontName = fontName;  
    }  
}
```


Interface et héritage

- Surcharge et annotation `@override`
 - Implémentation d'interface
 - Surcharge d'une méthode de la classe mère
- Pas d'héritage multiple en java mais possibilité d'implémenter plusieurs interfaces
 - 1 extends
 - plusieurs implements
- Exemple de masquage et surcharge

Exercice 7 - TextArea

- Au sein de la classe `TextArea`, écrire deux méthodes pour configurer la fonte de la zone de texte
 - Elle doit reprendre au moins deux des arguments de l'interface `FontTheme` (nom, taille, couleur)
- Masquage de la méthode *`String toString()`*
 - Doit afficher le texte de la zone de texte

Conventions de nommage – Valable pour les classes

- **Variable names** are case-sensitive. A variable's name can be any legal identifier — an unlimited-length sequence of Unicode letters and digits, beginning with a letter, the dollar sign "\$", or the underscore character "_". The convention, however, is **to always begin your variable names with a letter**, not "\$" or "_". Additionally, the dollar sign character, by convention, is never used at all. You may find some situations where auto-generated names will contain the dollar sign, but your variable names should always avoid using it. A similar convention exists for the underscore character; **while it's technically legal to begin your variable's name with "_", this practice is discouraged. White space is not permitted.**
- Subsequent characters may be letters, digits, dollar signs, or underscore characters. Conventions (and common sense) apply to this rule as well. When choosing a name for your variables, use full words instead of cryptic abbreviations. Doing so will make your code easier to read and understand. In many cases it will also make your code self-documenting; fields named cadence, speed, and gear, for example, are much more intuitive than abbreviated versions, such as s, c, and g. **Also keep in mind that the name you choose must not be a keyword or reserved word.**
- If the name you choose consists of only one word, **spell that word in all lowercase letters. If it consists of more than one word, capitalize the first letter of each subsequent word.** The names gearRatio and currentGear are prime examples of this convention. If your variable stores a **constant value**, such as `static final int NUM_GEARs = 6`, the convention **changes slightly, capitalizing every letter and separating subsequent words with the underscore character.** By convention, the underscore character is never used elsewhere.

Première lettre est une minuscule pour les variables

Première lettre est une majuscule pour les classes et interfaces

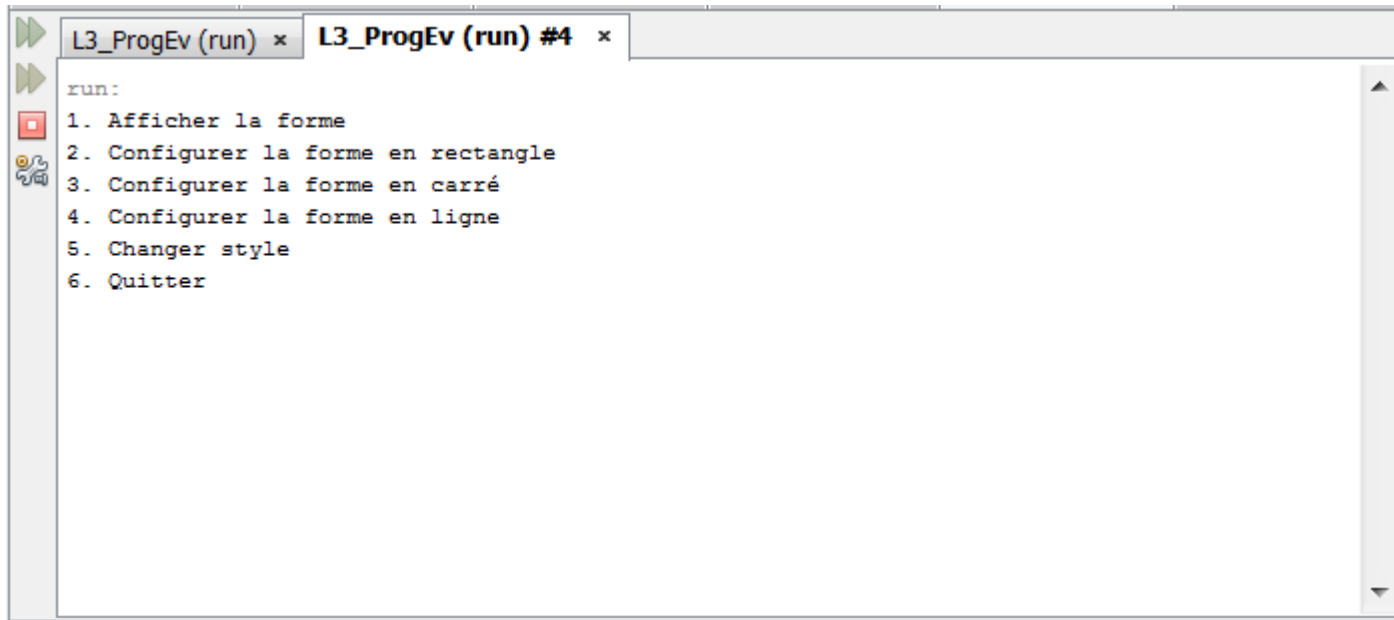
Anglais préférable

Exercice 8

- Ecrire un programme interactif qui selon le genre de l'utilisateur affiche "Bonjour Madame" ou "Bonjour Monsieur"

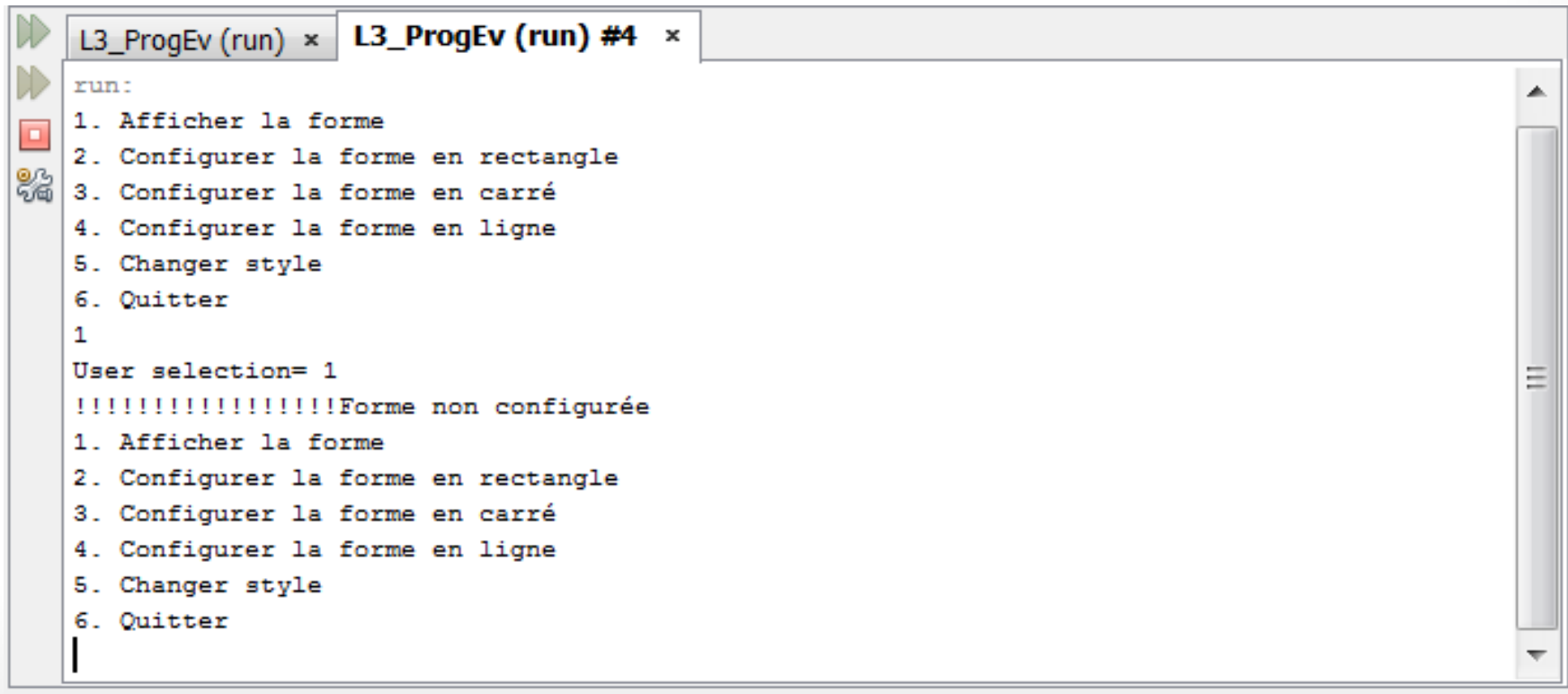
Exercice 9

- Ecrire un programme interactif à choix multiple



Exercice 9

- Ecrire un programme interactif à choix multiple

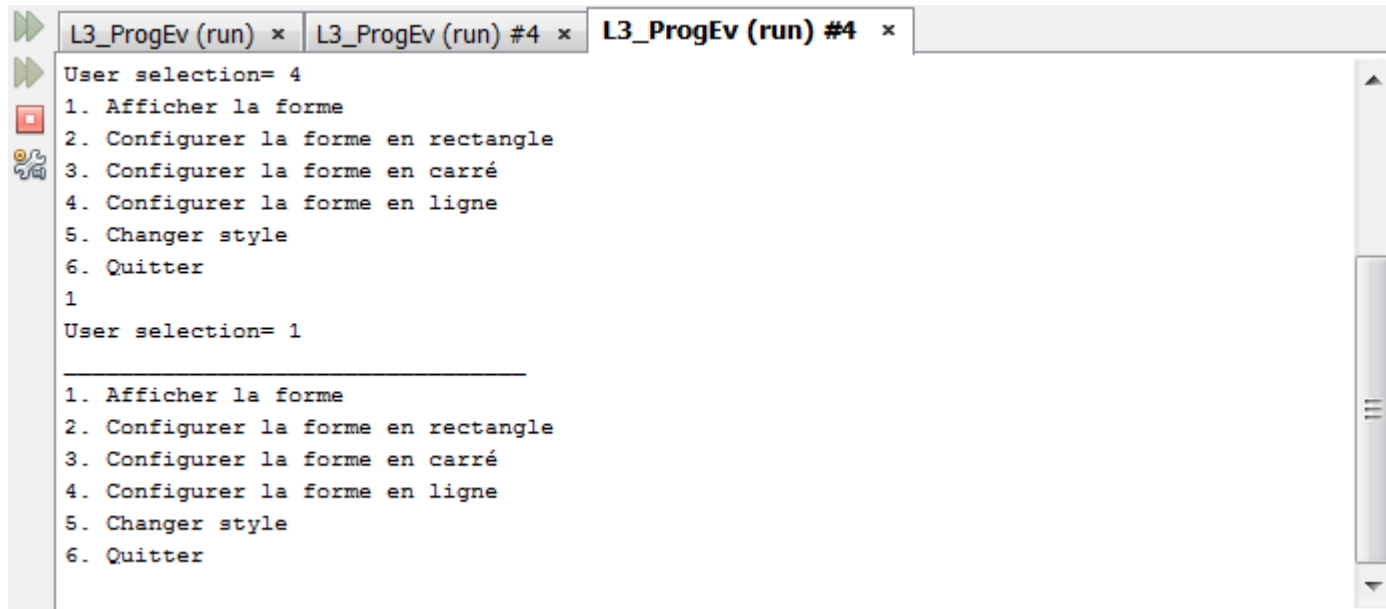


The screenshot shows a Python IDE with two tabs: 'L3_ProgEv (run)' and 'L3_ProgEv (run) #4'. The code in the active tab is a menu-driven program that displays a list of options and prompts the user to select one. The code is as follows:

```
run:
1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
1
User selection= 1
!!!!!!!!!!!!!!!!!!!!Forme non configurée
1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
|
```

Exercice 9

- Ecrire un programme interactif à choix multiple



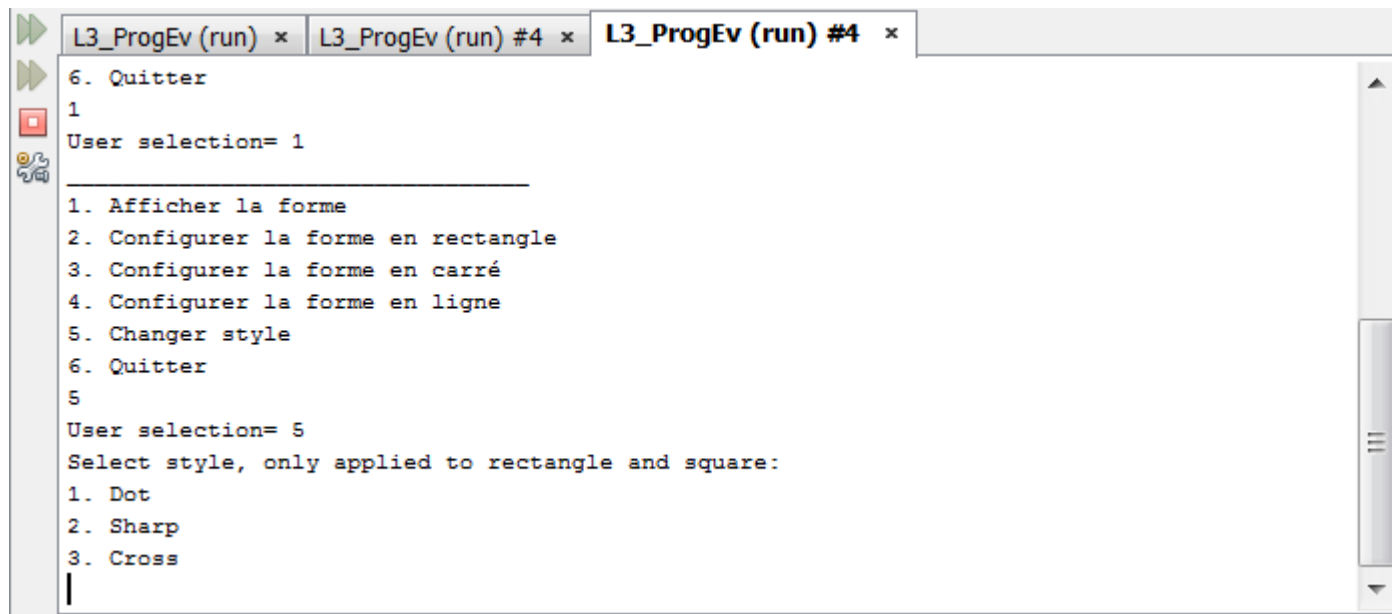
The screenshot shows a Python IDE with three tabs: 'L3_ProgEv (run) x', 'L3_ProgEv (run) #4 x', and 'L3_ProgEv (run) #4 x'. The active tab displays the following code:

```
User selection= 4
1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
1
User selection= 1

1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
```

Exercice 9

- Ecrire un programme interactif à choix multiple



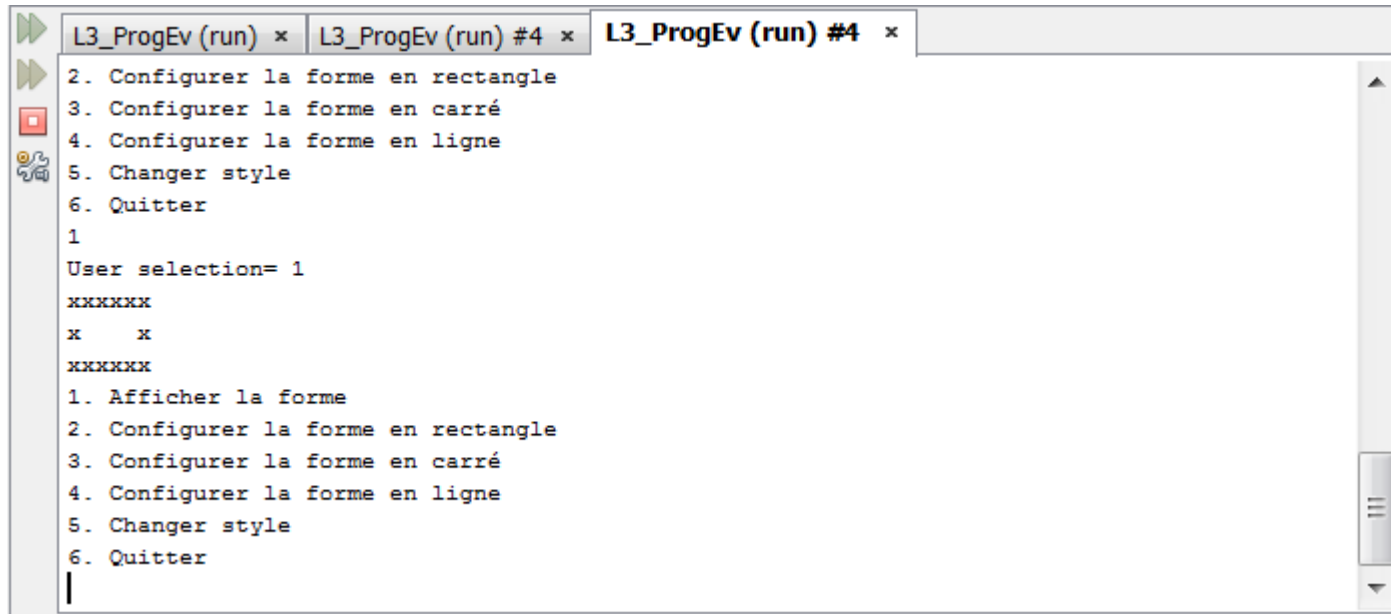
The screenshot shows a Python IDE with three tabs: 'L3_ProgEv (run) x', 'L3_ProgEv (run) #4 x', and 'L3_ProgEv (run) #4 x'. The active tab displays the following code:

```
6. Quitter
1
User selection= 1

1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
5
User selection= 5
Select style, only applied to rectangle and square:
1. Dot
2. Sharp
3. Cross
|
```


Exercice 9

- Ecrire un programme interactif à choix multiple



The screenshot shows a Python IDE with three tabs: 'L3_ProgEv (run) x', 'L3_ProgEv (run) #4 x', and 'L3_ProgEv (run) #4 x'. The active tab displays the following code:

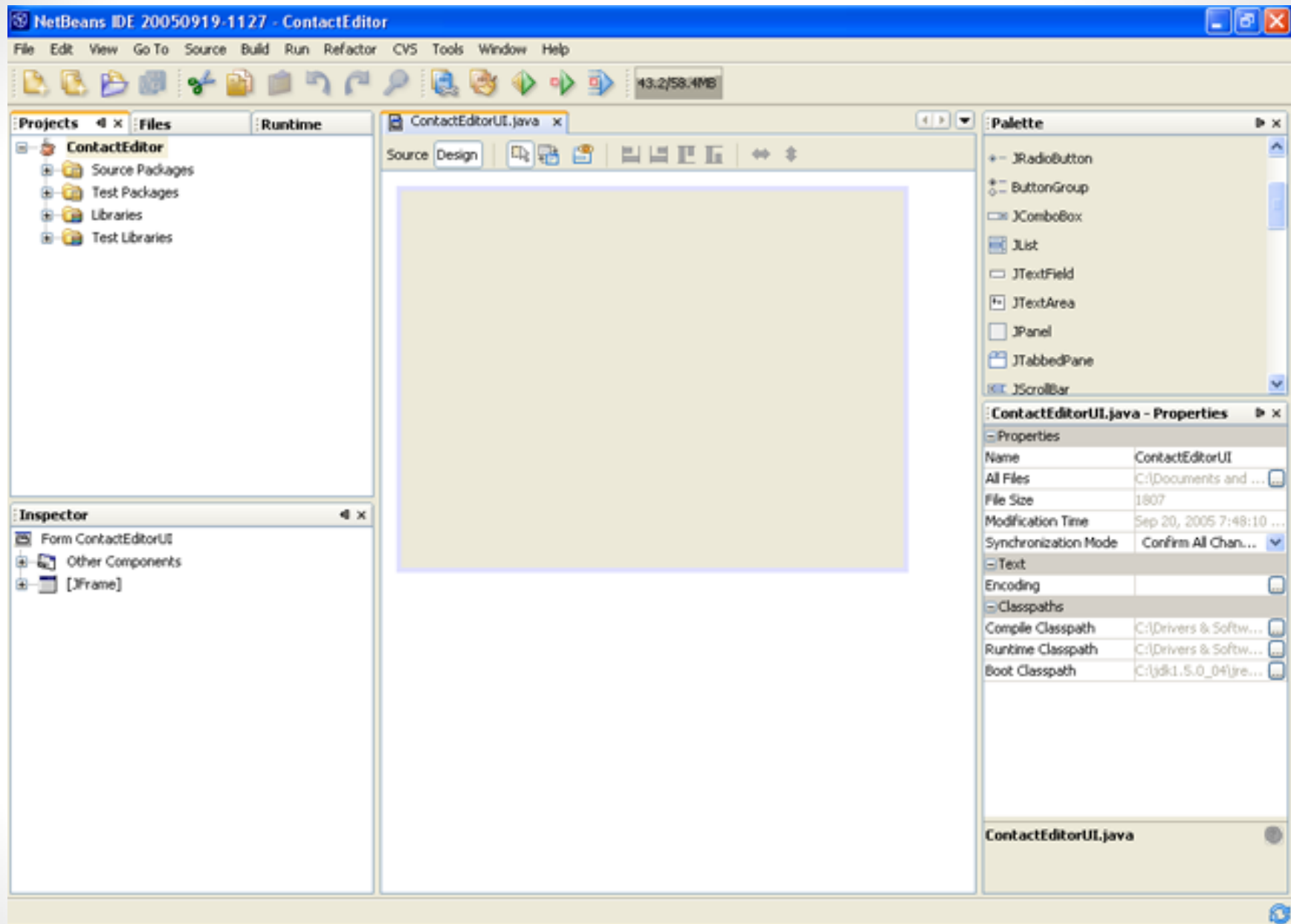
```
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
1
User selection= 1
xxxxxxx
x      x
xxxxxxx
1. Afficher la forme
2. Configurer la forme en rectangle
3. Configurer la forme en carré
4. Configurer la forme en ligne
5. Changer style
6. Quitter
|
```

Exercice 9

- Ecrire un programme interactif à choix multiple
 - Main et décomposition modulaire (méthodes d'affichage)
 - Hiérarchie de classe pour les différents types de formes
 - Héritage et surcharge
 - Spécification d'une interface pour la gestion des styles
 - Implémentation de l'interface

Netbeans

- <http://fr.netbeans.org/>
- Environnement de développement intégré (EDI)
- Support au développement d'interfaces graphiques



Netbeans

- <http://fr.netbeans.org/>
- Environnement de développement intégré (EDI)
- Support au développement d'interfaces graphique
 - Vue code source – “Source”
 - Vue conception graphique – “Design”
 - Palette de composants
 - Vue hiérarchique des composants d'une interface graphique
 - Vue propriétés d'un composant
 - Génération automatique de code
- Glisser-déposer, Feedback visuel

Démonstration

- Création d'un projet
- Création d'une fenêtre
- Utilisation des différentes vue de conception

COURS – TD 3: COMPOSANTS SIMPLES

RÉFÉRENCES

Internet

- <http://docs.oracle.com/javase/tutorial/index.html>

Ouvrages