CSI1502: Introduction au génie logiciel.

Chapitre 2:
Concepts de base de Java

Objectifs du cours: Concepts de base de Java

- Présentation des objets et de leurs propriétés.
  - Objets prédéfinis: System.out, String
- Variables, affectations.
- Types de données primitifs
  - Définition des 8 types.
  - Manipulation d'expressions arithmétiques: opérateur de précédence et conversion de données.
- Création d'Objets en java (réutilisation de l'exemple 'String')
- Bibliothèques de classes et packages: Random, Math, Keyboard, etc...
- Introduction aux applets Java.

Introduction aux objets

- Un *objet* représente une entité pouvant interagir à l'intérieur d'un programme.
- Un objet fournit un ensemble de services que nous pouvons lui demander d'effectuer.
- Les services sont **définis par des méthodes** dans une ***classe***, elle même définissant l'objet.
- **Une classe représente un concept, et un objet une instanciation de classe**.
- Une classe peut donc être utilisée pour créer de multiples objets.

Un objet prédéfini:

- L'objet System.out représente une destination vers laquelle nous pouvons envoyer un résultat.
- Un service nous est rendu.
- Dans le programme Hello, nous invoquons la méthode 'println' de l'objet System.out:

```
System.out.println(« Hello world. My name is Suzy »);
```

Différence entre les méthodes *print* et *println*.

```
// countdown.java (p.65)
// Montrer la différence entre println et print

public class countdown {
// Sort les lignes représentant le compte à rebours d'une fusée
public static void main(String [] args) {
   System.out.print("Three...");
   System.out.print("Two...");
   System.out.print("One...");
   System.out.print("Zero...");

   System.out.println("Liftoff!");
   System.out.println("Houston we have a problem.");
   }
```

Différence entre les méthodes *print* et *println*.

- La **méthode** *print* est similaire à la **méthode** *println*, cependant elle ne fait pas de retour a la ligne.
- Ces deux **méthodes** font partie de **l'objet** System.out .
- Les **méthodes** sont invoquées au moyen de **paramètres**, ces derniers étant des messages envoyés à la méthode.

Que sont les classes et les objets?

- Une classe (un concept): Compte banquaire.
- Plusieurs objets (instanciations) de la même classe:
  - Compte banquaire de Jacques: 11000 $
  - Compte banquaire de Jean: 1000 $
  - Compte banquaire de Gilles: 100 $

Définitions de la fonction d'objet.

- Les objets ont été introduits principalement comme structure de données conceptuellement autonomes et indépendantes.
- Objet: Serveur de restaurant.
- Actions: Communiquer avec soi-même (?!?), les clients et le personnel.
- Un objet est une représentation abstraite d'une entité réelle. Cette représentation a une identité unique, des propriétés intrinsèques. Un objet peut communiquer avec lui-même ainsi qu'avec d'autres objets.

Propriété d'objets (1): Héritage

- Une classe peut-être dérivée d'une autre grâce au mécanisme d'*héritage*.
- Les classes peuvent être organisées hierarchiquement selon leurs relations d'héritage.

Propriété d'objets (2): Abstraction

- Le mécanisme d'*abstraction* permet de cacher le bon détail au moment adéquat.
  - -> les détails internes sont cachés.
- Un objet est une entité abstraite dans la mesure où l'on peut l'utiliser sans en connaître les détails internes (**boîte noire**).
- Par exemple nous pouvons utiliser la méthode println sans en connaître l'implémentation.

Propriété d'objets (2): Abstraction

- Pourquoi utiliser un mécanisme d'abstraction?
  - Un être humain en peut gérer que 7 (plus ou moins 2) informations au même moment.
  - Cependant si nous regroupons l'information en morceaux (objets), il devient possible de manipuler plus d'informations en même temps.
- Les classes et les objets nous aident à créer des logiciels complexes.

Exemple d'objet prédéfinis: Strings (voir Facts.java p.68)

```
// Montrer l'utilisation de la concaténation de chaînes
   de caractères et de la conversion automatique.
public class Facts {
   // imprime "good to know":
   public static vodi main(String []argv) {
    // Concatène deux chaînes en une:
    System.out.println(«We present the following facts
for use »);
    // Une chaîne contenant un nombre:
    System.out.println(«Numbers in Hawaiian
alphabet:12 »);
```

Exemple d'objet prédéfinis: Strings

- Chaque chaîne de caractères est un objet en Java, défini par la classe String.
- Chaque chaîne délimitée par des guillemets (“”), représente un objet String.
- L'opérateur de concaténation (+) est utilisé pour mettre deux chaînes l'une à la suite de l'autre.
- Cet opérateur peut aussi être utilisé pour faire suivre une chaîne par un nombre.
- **Une chaîne de caractères ne peut être définies sur deux lignes dans un programme**

Concaténation et addition arithmétique:

```
// Addition.java
// Montre la différence entre la concaténation de chaînes et l'addition
   arithmétique:
   public class Addition {
   // Concaténation de deux nombre. Addition de deux nombres et sortie
   du résultat.
   public static void main(String[] argv) {
    System.out.println("24 et 60 concatenated: "+24+60);
    System.out.println("24 et 60 added: "+(24+60));
   }
}
```

Concaténation et addition arithmétique:

- L'opérateur + est aussi utilisé pour l'addition arithmétique.
- La fonction opérée par l'opérateur + dépend du type de donnée qui lui est passé.
- Si **les deux ou l'une des opérandes** sont des chaînes, l'opération sera une **concaténation**.
- Si les **deux** opérandes sont des **nombres**, ce sera une **addition** arithmétique.
- L'opérateur + est évalué de gauche à droite.
- Les parenthèses peuvent être utilisées pour

Chaînes de caractères: séquences d'échappement.

- Comment imprimer un guillemet dans une chaîne?
- La ligne suivante créerait une confusion au niveau du compilateur car le second guillement serait interprété comme la fin de la chaîne.
  - System.out.println(“I said “Hello” to you.”);
- Une séquence d'échappement est une suite de caractère représentant un caractère spécial.
- Une séquence d'échappement commence avec un backslash (\), indicant que le

Séquences d'échappement: suite...

- Quelques séquences d'échappement :
  - \b => backspace
  - \t => tabulation
  - \n => nouvelle ligne
  - \r => retour chariot
  - \” => guillemet
  - \’ => apostrophe
  - \\ => backslash
- Voir Roses.java (page 71)

Utiliser des variables en Java

- Une *variable* est un nom pour une case mémoire.
- Une variable doit être déclarée en donnant le nom de la variable et en indicant le type de donnée qui sera contenu:
  - int total;
- int => type de donnée.
- total => nom de variable.
- Plusieurs variables peuvent être crées en une déclaration:
  - int count, temp, result;

Variables et affectations

```
// Geometry.java (affectation de variables)
 public class Geometry {
    public static void main(String []argv) {
     int sides = 7;
     System.out.println("A heptagone has "+ sides + "sides");
     sides = 12;
     System.out.println("A dodegan has "+ sides + "sides");
    }
}
```

Variables et affectations:

- Une valeur intiale peut être donnée à la variable lors de la déclaration:
  - int sum = 0;
- Quand une variable est référencée dans un programme, sa valeur courante est utilisée.
- Une *affectation* change la valeur d'une variable.
- Vous ne pouvez donner à une variable qu'une valeur cohérente avec le type de la variable.

Constantes:

- Une constante est une case mémoire identifiée comme une variable n'ayant **qu'une seule valeur** tout au long de la durée de vie du programme.
- Le compilateur donnera une erreur si vous essayez d'en changer la valeur.
- Le mot-clé utilisé pour déclarer une constant en Java est *final*.
  - final int MIN_HEIGHT = 69;
- Les constantes:
  - nomment des variables numériques.
  - facilitent les mises à jour de valeurs au cours d'un programme.
  - empêchent de changer certaines valeurs par inadvertance.

Types de données primitifs:

- Il y a exactement 8 types de donnée primitifs en Java:
- 4 d'entre eux représentent des entiers:
  - byte, short, int, long.
- 2 d'entre eux des nombres en virgule flottante:
  - float, double.
- 1 des caractères:
  - char.
- 1 des valeurs booléennes.
  - boolean.

Types de données primitifs (1-6):

- La différence entre les différents types primitifs de données numériques réside dans leur taille et donc dans les valeurs qu'ils peuvent stocker:
  - byte | 8 bits | -128 | 127.
  - short | 16 bits | -32,768 | 32,767.
  - int | 32 bits | -2,147,483,648 | 2,147,483,647.
  - long | 64 bits | < $-9*10^{18}$ | > $9*10^{18}$
  - float | 32 bits | +/- 3.4 $*10^{38}$, avec 7 chiffres significatifs.
  - double | 64 bits | +/- 1.7 $*10^{308}$, avec 15 chiffres significatifs.

Types de données primitifs 7: Les caractères

- Une variable *char* stocke un unique caractère de l'ensemble de caractères *Unicode*.
- Un ensemble de caractères est une liste ordonnée de caractères dont chaque caractère correspond à un nombre de manière unique.
- L'ensemble de caractères Unicode utilise 16 bits par caractère, permettant l'indexation de 65,536 caractères distincts.
- Il s'agit d'un ensemble international, comprenant des symboles et des caractères de différentes langues.
- Les nombre affectés aux caractères peuvent être obtenus en délimitant un caractère par deux apostrophes:
  - 'b'; '?' ; 'L' ; '\n'

Types de données primitifs 7: Les caractères

- L'ensemble de caractères ASCII est plus vieux et plus petit que Unicode, cependant est toujours utilisé.
- Les caractères ASCII sont un sous ensemble de l'ensemble Unicode, comprenant:
  - Les majuscules
  - Les minuscules
  - Les caracères de ponctuation
  - Les chiffres
  - Les symboles spéciaux &, | , / ....
  - Les caractères de contrôle (retour à la ligne, tabulation...)

Types de données primitifs _: Les booléens

- Une valeur booléenne équivaut à un *vrai* ou à un *faux .*
- Une valeur booléenne peut être utilisée pour représenter tout état ne pouvant avoir que deux valeurs
  - Une lampe allumée ou éteinte.
- Les mots clé *true* et *false* sont les seules valeurs des variables booléenes.
  - Boolean done = false ;

Types de données primitifs: les expressions arithmétiques.

- Une *expression* est une combinaison d'un ou plusieurs opérandes et de leur opérateurs.
- Une expression arithmétique calcule un résultat numérique et utilise les opérateurs suivants:
  - Addition +
  - Soustraction –
  - Multiplication *
  - Division /
  - Congruence %
- Si l'un ou deux des opérandes sont des nombres flottants, le résultat sera aussi un nombre flottant.

Division et reste...

- Si les deux opérandes de l'opérateur / sont des entiers, les resultat est un entier (la partie fractionnaire est ignorée).
  - 14 / 3 ?
  - 8 / 12 ?
- L'opérateur de congruence % retoune le reste après avoir diviser le premier opérande par le second.
  - 14 % 3 ?
  - 8 % 12 ?

Type primitifs: règles de précédence des opérateurs.

- Les opérateurs peuvent être combinés dans des expressions complexes.
  - Result = total + count / max – offset;
- Les règles de précédence donnent l'ordre dans lequel les différentes opérations sont effectuées:
  - *, /, % sont effectués avant +, -, et la concaténation de chaînes.
  - Les opérateurs ayant une même priorité sont évalués de gauche à droite.
  - Les parenthèses peuvent être utilisées pout forcer l'ordre d'évaluation.

Précédence des opérateurs

- Donnez l'ordre d'évaluation des opérateurs dans les expressions suivantes:
  - a+b+c+d+e a+b*c/e
  - a/(b+c) d%e
  - a/(b*(c+(d e)))

Affectation: suite

- L'opérateur d'affectation a une priorité inférieure à celles des opérateurs arithmétiques.
  - answer = sum/4+MAX*lowest;
  - L'expression arithmétique est évaluée puis stockée dans *answer*.

Affectation: suite

- Les deux opérandes de l'opérateur d'affectation peuvent contenir la même variable:
  - count = count + 1;
  - Un est d'abord ajouté à la valeur courante de *count*.
  - Puis le résultat est stocké dans *count*, effaçant la valeur originale.

Conversion de données

- Nous voudrions par exemple convertir un **entier** en un **nombre flottant** pour un calcul.
- Les conversions doivent être considérées avec précautions pour éviter des pertes d'information.
- En Java les conversions peuvent se faire de 3 façons:
  - Conversion par affectation.
  - Conversion pqr promotion arithmétique
  - Conversion par *casting.*

Conversions: agrandir/diminuer la précision.

Byte	Short, int,long,float ou double
Short	Int,long,float ou double
Char	Int, long, floqt ou double
Int	Long,floqt ou double
Long	Float ou double
float	double

Byte	char
Short	Byte ou char
Char	Byte ou short
Int	Byte, short ou char
Long	Byte, short, char ou int
float	Byte, short, char,int ou long
double	Byte;short;char, int, long ou float

Conversions de donnéesm

- **Conversion par affectation**: quand la valeur d'une variable est affectée à une variable d'un autre type.
  - Seules les conversions augmentant la précision peuvent être faites par affectation.
  - money = dollars; // int en float
- **Conversion par promotion** arithmétique se produisent automatiquement quand les opérateurs arithmétiques convertissent leurs opérandes.
  - result = sum / count; // float = float / int.

Conversions de données:

- Le *casting* est la plus puissante, dangereuse, technique de conversion des données.
  - L'augmentation, diminution de précision peuvent être accomplies par casting.
  - Le casting se fait en affichant le nouveau type de la variable entre parenthèses devant celle-ci.
- Par exemple si total et count sont des entiers, que nous desirons un résultat flottant à l'issue de la division, nous pouvons *caster* total:
  - result = (float) total / count;

Casting: exemple

```
public class Casting {
    public static void main(String []argv) {
      int x = 10, y=4;
      int a;
      float b,c,d,e;
      float z =10;

      a = x / y;
      b = x / y;
      c = (float) x / y;
      d = (float) x / y;
      e = z / y;

      System.out.println("Integer division result:"+a);
      System.out.println("Floating division result:"+b);
      System.out.println("Floating casting division result:"+c);
      System.out.println("Floating ans parenthesis result:"+d);
      System.out.println("Arithmetic promotion:"+e);
    }
}
```

Casting: résultats de l'exemple

- Integer Division result:2
- Floating Division result:2.0
- Float casting Division Result:2.5
- Float and parenthesis Result/2.0
- Arithmetic promotion:2.5
- Les déductions:
  - Si l'on ne veut pas perdre de précision, nous devrions:
    - Definir nos nombres en virgule flottante (sûr et simple).
    - Utiliser le casting.

Création d'objets:

- Une variable contient soit un **type primitif de donnée** ou **une référence à un objet**.
- Un nom de classe peut être utilisé pour déclarer une variable référant à un objet de cette classe:
  - String title;
- **Aucun objet n'est créé avec une telle déclaration**.
- Une variable référant à un objet contient **l'adresse** de l'objet.
- L'objet en soi **doit être créé séparemment**.

Créer des objets:

- La création d'un objet s'effectue au moyen de l'opérateur *new*.
  - Title = new String("Java Software Solution");
  - L'opérateur *new* appelle le constructeur de la classe String prenant en argument une chaîne de caractères.
- La création d'un objet est aussi appelée *instanciation de l'objet*.
- Un objet, rappelons le, est une instance d'une certaine classe.

Créer des objets:

- Les chaînes de caractères étant des objets très courants, l'utilisation de l'opérateur new est facultative.
  - title = "Java Software Solution";
- Ceci est uniquement valable pour les objets de la classe String.
- Après l'instanciation d'un objet, l'invocation de ses méthodes se fait à l'aide de l'opérateur point (.).
  - title.length();
  - System.out.println();

Les méthodes de la classe String

- La classe String offre plusieurs méthodes utiles pour la manipulation de chaînes de caractères.
  - length;
  - toLowerCase;
  - Substring;
- Voir la page 89 et l'appendice M du livre de cours pour les autres méthodes.
- De nombreuses méthodes *retournent une valeur* (un entier ou un nouvel objet String).

Un exemple de méthode de la classe String: Substring

```
public class TestSubstring {
   // Soyez sûr de comprendre le fonctionnement de 'Substring'.
   public static void main(String []argv) {
   String phrase = "Hello there world";
   String mutation1;
   mutation1 = phrase.substring(3,13);
   System.out.println("The result is : "+mutation1);
   System.out.println("The string length is :"+mutation1.length());
   }
}
```

- The result is : lo there w
- The string length is : 10

Un autre exemple utilisant les méthodes de la classe String.

```
public class StringMutation {
   public static void main(String []argv) {
   String phrase = new String("Change in unavoidable");
   String mutation1, mutation2, mutation3,mutation4;
   mutation1 = phrase.concat(", except for some machines.");
   mutation2 = mutation1.toUpperCase();
   mutation3 = mutation2.replace('O', 'Z');
   mutation4 = mutation3.substring(3,12);
   System.out.println("Final string: "+ mutation4);
   System.out.println("Final string length: "+mutation4.length);
   }
}
```

- Final string: NGE IN UN;
- Final string length: 9;

Les bibliothèques de classe Java

- Une *bibliothèque de classes* est un ensemble de classe pouvant être utilisées pour écrire un programme.
- La *bibliothèque standart Java* fait partie de tout environnement de programmation Java.
- Les classes ne font pas partie en soi du langage Java mais sont utilisées de manière extensive.
- Les classes *System* et *String* en font partie.
- Les autres classes peuvent être fournies par des organisations, ou vous pouvez les créer vous-même.

Les bibliothèques de classe Java

- Les classes de la *bibliothèque standart Java* sont organisées en **packages**, ensembles de classes ayant un caractère commun.
- Un groupe de classes apparentées est appelé une API Java (Application Programming Interface).
  - **i.e.** l'API Swing, l'API Java de base de données.
- Une API donnée peut contenir plusieurs packages.
- Voir **Appendice M** du livre de cours.

Les packages

- Certains des packages de la bibliothèque standart Java sont:
  - java.lang=>fonctionnalités générales.
  - java.applet=>création d'applets pour le web.
  - java.awt =>interfaces utilisateurs et dessins.
  - javax.swing =>extension graphiques
  - java.net =>communication réseau.
  - java.util => outils divers.
  - java.xml.parsers =>traitement de documents xml
- Voir Appendice M.

Le mot clé *import*

- Pour utiliser une classe d'un package, vous pouvez la nommer complétement:
  - java.util.Random
- Une autre possibilité consiste à importer la classe et à utiliser son nom:
  - import java.util.Random;
- Pour importer toutes les classes d'un package, vous pouvez utiliser le caractère étoile (*):
  - import java.util.*;

Le mot clé *import*

- Toutes les classes du package *java.lang* sont importées par défaut dans tous les programmes.
- C'est pourquoi nous n'avons pas eu à importer les classes *System* et *String* dans nos programmes précédents.
  - La classe Random fait partie du package java.util .
  - Elle fournit des méthodes pour générer aléatoirement des nombres.

Exemple: import de la classe Random

```
import java.util.Random;
public void RandomNumbers {
   public static void main(String []argv) {
     // generator est un objet de la classe Random
     Random generator = new Random();
     int num1;
     num1 = generator.nextInt();
     System.out.println("A random integer: "+num1);
     num1 = generator.nextInt(15);
     System.out.println("From 0 to 14: "+num1);
   }
}
```

Un exemple: la classe Random

- Première exécution:
  - A random number: -682994049
  - From 0 to 14: 3
- Deuxième exécution:
  - A random number: -908887626
  - From 0 to 14: 2
- Troisième exécution:
  - A random number: 1900056825
  - From 0 to 14: 9

Méthodes *de classel statiques*.

- Certaines méthodes peuvent être invoquées directement à partir **du nom de la classe** et **non à partir d'un objet de la classe**.
- Ces méthodes sont appelées *méthodes de classe* ou *méthodes statiques*.
- La classe Math (voir 2.13 page 99) contient plusieurs méthodes statiques, fournissant différentes fonctions mathématiques (valeur absolue, fonctions trigonométriques, racine carrée,etc...).
  - temp = Math.cos(90) + Math.sqrt(delta);

La classe Keyboard

- La classe keyboard ne fait pas partie de la bibliothèque standart Java. Elle est fournit par les auteurs du libre de cours pour faciliter la lecture des caractères tapés à l'aide du clavier.
- L'objet System.in fait partie de la bibliothèque standart Java (discussion chapitre 8).
- Les détails de la classe Keyboard sont exposés au chapitre 5.
- La classe keyboard fait partie du package *cs1*.
- Elle contient plusieurs méthodes statiques pour lire différents types de données.
- Voir aussi Quadratic.java (page 102)

Exemple: Classe Keyboard

```
import cs1.Keyboard;
public class Echo {
   // lit ce que l'utilisateur a tapé et l'affiche
   public static void main(String []args) {
     String message;
     int a;
     System.out.println("Enter any text:");
     message = Keyboard.readString();
     System.out.println("Enter any integer: ");
     a = Keyboard.readInt();
     System.out.println("You entered: \""+message+\""+ " and "+a);
   }
}
```

- You entered: "Hello world" and 23

Formater les résultats en sortie

- La classe DecimalFormat peut être utilisée our formater un nombre en virgule flottante de différentes manières.
- Vous pouvez par exemple préciser que le nombre doit être sorti avec 3 décimales.
- Le constructeur de la class DecimalFormat prend une chaîne de caractères en argument, représentant un exemple de nombre formaté.
- Voir CircleStats.java (page 107)
- Voir aussi la classe NumberFormat page 103.

Introduction aux applets

- Une **application** Java est un **programme autosuffisant** avec une méthode main (comme celles que nous avons vues jusqu'à présent).
- Un applet Java est un programme conçu pour pouvoir être transmis sur le web et exécuté dans un navigateur.
- Un applet peut aussi être exécuter par l'intermédiare d'un *applet viewer* fourni dans le kit de developpement Java.
- **Un applet n'a pas de méthode** *main*.
- Un applet comprend à la place plusieurs méthodes particulières déservant des buts précis.

Exemple d'applet: Einstein.java

```
import java.Applet;
import java.awt.*;
public class Einstein extends Applet {
   // dessine une citation et quelques formes géométriques.
   public void paint(Graphics page) {
    page.drawRect(50,50,40,40); // carré
    page.drawOval(75,65,20,20); // cercle
    page.drawString("Imagination is more important than
  knowledge.",110,70);
    page.drawString("-Albert Einstein",130,100);
   }
}
```

Discussion au sujet de l'applet Einstein.java

- Une classe définissant un applet *étend* la classe Applet (***extends***).
  - Nous faisons usage de l'héritage présenté au chapitre 7.
- La méthode paint() est exécutée automatiquement et est utilisée pour dessiner le contenu de l'applet.
- Elle prend un **paramètre** qui est un objet de la classe **Graphics**.
- Un objet **Graphics** définit un contexte graphique sur lequel des figures géométriques ainsi que des textes peuvent être dessinés.
- La classe Graphics contient plusieurs méthodes de dessin de figures géométriques.

Comment sont exécutés les applets?

- Run Applet(jGrasp; Realj)
- Un applet est incorporé au contenu d'un fichier HTML en utilisant un marqueur qui renverra au code binaire de l'applet.
- Le code binaire du programme est communiqué à travers le web et exécuté par un interpréteur Java faisant partie d'un navigateur.
- Voir appendice J pour plus d'informations.

Marqueur HTML d'un applet

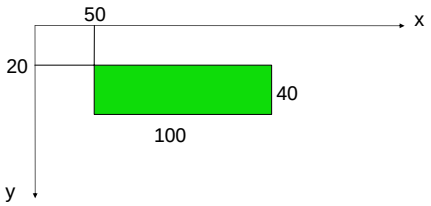
```
<html>
   <head>
    <title>The Einstein Applet</title>
   </head>
   <body>
    <applet code="Einstein.class" width =350 height=175>
    </applet>
   </body>
</html>
```

Dessiner des figures géométriques:

- La class Graphics peut dessiner des figures géométriques avec plus de détails:
  - Une figure peut être coloriée selon la méthode appelée.
  - Les paramètres spécifient les coordonnées et les dimensions.
  - Chapitre 1 => le système de coordonnées Java a son origine en haut à gauche.
  - Les figures courbes comme le cercle sont généralement dessinées en spécifiant le rectangle dans lequelles elles sont inscrites.
  - Voir 2.18 page 112 pour des méthodes.

Exemple: dessin d'un rectangle

```
page.SetColor(Color.Green);
page.fillRect(50,20,100,40);
```



La classe Color

- Une couleur peut être définie dans un programme Java en utilisant un objet de la classe Color.
- Cette classe contient aussi plusieurs couleurs prédéfinies, disponibles en constantes statiques.
  - Color.black Valeurs RGB: 0,0,0
  - Color.blue Valeurs RGB: 0,0,255
  - Color.cyan Valeurs RGB: 0,255,255
  - Color.orange Valeurs RGB: 255,200,0
  - Color.white Valeurs RGB: 255,255,255
  - Color.yellow Valeurs RGB: 255,255,0

Chapitre 2: Résumé

- Objets prédéfinis.
- Types de donnéees primitifs.
- Déclaration et utilisation des variables.
- Expressions et priorités des opérateurs.
- Créer et utiliser des objets.
- Bibliothèques de classes.
- Applets Java.
- Dessiner des figures géométriques.