JAVA et netbeans

Nouveau projet :

File/New Project
dans **catégories** : java dans **projects** : java application

Nouvelle classe :

File/New file
dans **catégories** : java dans **project** : Java Class

Nouveau Packages :

Cree un package :

clic droit sur source package / new / java package
nom du package com.sdz.test (lien a l'envers + nom)
clic droit sur le nouveau package / new / Java class

Pour appeler le fichier : import com.sdz.test.Ville; (ville = nom du fichier dans package)

Penser à le mettre **public** et pas **static**

// /* */	commentaires

package projet1;
 	
/**
*@author dalan
**/
public class Projet1 {
 	
public static void main(String[] args) {
 	
}
}
	Ce qui est censé avoir avant de commencer a coder

System.out.print("Hello World !"); System.out.println("Hello World !");	Affiche Hello World ! dans la console Affiche Hello World ! dans la console, le prochain sera écrit a la ligne suivante (ou \n pour aller à la ligne et \t pour faire une tabulation)

try {
 	
}
catch (nomExeption e) {
 	
}
	Exécute le code de try
	
	Si il y a une erreur, exécute catch à la place

Variables:

byte nom_variableEntier = 64;
short nom_variableEntier= 3200;
int nom_variableEntier= 15600000;
long nom_variableEntier= 94607000000000000L;
float nom_variableFloat= 9.141592653f;
char nom_variableChar= "a";
boolean nom_variableBool = True;
String phrase = "coucou";
	-128 et +127.
	-32768 et +32767.
	-2*109 à 2*109
	−9×1018 à 9×1018 (L a la fin : 94607000000000000**L**)
	Mettre un f à la fin (9.0f)
	Une seule lettre
	True ou False rien d'autre
	Une chaine de caractère ("abcdef…0123…") **attention à la majuscule a String**

nbre1 += 1; nbre = nbre + 1;
nbre1 -= 1; nbre = nbre + 1;
nbre1 *= 1; nbre = nbre + 1;
nbre1 /= 1; nbre = nbre + 1;
	+1
	-1
	*1
	/1

(int)variable (float) variable string = string.valueOf(entier) Integer.valueOf(string).intValue();	Convertit en int Convertit en float Convertit un int en string Convertit un string en int
Scanner sc = new Scanner(System.in);	**A mettre qu'une fois pour qu'on puisse entrer une valeur**
int str = sc.nextInt();	Demande à entrer une valeur **Fonctionne avec tout ce qui est chiffre (nextInt, nextFloat…)** Pour les chaines de caractère
String str = nextLine(); sc.nextLine();	Demande à entrer une chaine de caractère **Mettre cette ligne à chaque demande de chaine**
.length()	Renvoie la taille d'un string
.toLowerCase()	Convertit en minuscule
. toUpperCase()	Convertit en majuscule
str1.equals(str2)	Vérifie si str1 == str2 **(pour les string)**
str.charAt(i)	Renvoie le caractère à l'emplacement i de la chaine str
.substring(3,13)	Renvoie les 4,5,6,7,8,9,10,11,12,13 èmes caractères de la chaine
.indexOf('t')	Retourne la position de la lettre (ou mot) t
Math.random(); Math.sin(120); Math.cos(120); Math.tan(120); Math.abs(-120.25); Math.pow(d, 2);	Chiffre aléatoire entre 0 et 1 (0.0001385746329371058) Sinus Cosinus Tangente Valeur absolue Exposant
Math.random()*100;	Chiffre aléatoire entre 0 et 100**(non inclus)**
int random = (int)(Math.random() * (nbA- nbB)) + nbB;	Nombre compris entre nbA et nbB **(A plus grand que B)**

Conditions:

if (varable < 0)
{
 	
}
else if(varable > 0)
{
 	
}
else
{
 	
}
	SI
	
	SINON SI
	
	SINON

switch (note)
{
case 0:
//code
break;
case 10:
//code
break;
case 20:
//code
break;
default:
//code
}
	Variable note
	
	SI note == 0
	
	SINON SI note == 10
	
	SINON SI note == 20
	
	SINON

while (/* Condition */)
{
 	
}
	TANT QUE

for(int i = 1; i <= 10; i++)
{
 	
}
	POUR

Tableau:

int tableau[] = new int[6] int tableau[][] = new int[6][6]	Tableau Tableau a 2 dimension
premiersNombres.length	Renvoie la taille du tableau
String tab[] = {"toto", "titi", "tutu", "tete", "tata"}; for(String str : tab)	Tableau de string Pour str dans tab

liste:

ArrayList maListe = new ArrayList();	Cree une liste maListe
maListe.add(maValeur);	Ajoute une valeur maValeur
maListe.get(5)	Retourne la valeur du 5eme element
maListe.remove(5)	Supprime le 5eme element
maListe.isEmpty()	Retourne True si la liste est vide
maListe.removeAll	Efface tout sans exceptions
maListe.contains("string")	Retourne true si String est dans la liste
al.get(1)	Lit la 1ere valeur de la liste

Dictionnaire :

Hashtable monDico = new Hashtable();	Cree un dictionnaire monDico
monDico.put(1, "printemps");	Ajoute 1 : printemps a monDico
monDico.isEmpty()	retourne « vrai » si l'objet est vide ;
monDico.contains(valeur)	retourne « vrai » si la valeur est présente
monDico.containsKey(valeur)	retourne « vrai » si la clé passée en paramètre est présente dans la Hashtable ;
monDico.elements()	retourne une énumération des éléments de l'objet ;
monDico.keys()	retourne la liste des clés sous forme d'énumération.

Les fonctions définies : A METTRE EN DEHORS DU MAIN

public static double arrondi(double A, int B) {
 	
return (double) ((int) (A * Math.pow(10, B) + .5)) / Math.pow(10, B);
}
	Cree une fonction arrondi qui aura un retour de type double
	Retourne un double

Arrondi(10.5,5)	Appelle la fonction arrondi avec les argument 10.5 et 5

Les class : AVANT LE MAIN

public class Ville {	Cree une Ville
private String nomVille; protected String nomPays;	Stocke nomVille en privé (seulement dans cette objet) Stocke nomPays en protected(si on veut faire un héritage, mettre protected)
public int nbreHabitants;	Stocke nbreHabitants en plublic **(Déconseillé sauf réél utilité)**

public **Ville**(){
 	
System.out.println("Création d'une ville !");
nomVille = "Inconnu";
nomPays = "Inconnu";
nbreHabitants = 0;
}
	Constructeur par défaut (dès que l'on crée sans donner de paramètre)
	Envoie le message création d'une ville
	Initialise nomVille
	Initialise nomPays
	initialise nbrehabitant

public **Ville**(String pNom, int pNbre, String pPays)
{
System.out.println("Création d'une ville avec des paramètres !");
nomVille = pNom;
nomPays = pPays;
nbreHabitants = pNbre;
}
	Constructeur avec des Paramètre (quand on crée en donnant paramètres, attention prend le même nom)
	Envoie le message Création d'une ville avec des paramètres !
	nomVille prend la valeur pNom
	nomPays prend la valeur pPAys
	nbreHabitant prend la valeur pNbre

public String getNom() {
return nomVille;
}
	Pour retourner le nom de la ville

public String getNomPays() {
return nomPays;
}
	Pour retourner le nom du pays

public int getNombreHabitants() {
return nbreHabitants;
}
	Pour retourner le nombre d'habitant

public String getAffiche(){
return "nom pays : "+this.getNom()+"\nnombre d\'habitant :"+this.getNombreHabitants()+"\nnom ville : "+this.getNom();
}
	Pour retourner tout d'un coup
	this = nom objet qu'on utilise

public void setNom(String pNom) {
nomVille = pNom;
}
	Pour changer le nom de la ville

public void setNomPays(String pPays)
{
nomPays = pPays;
}
	Pour changer le nom du pays

public void setNombreHabitants(int nbre)	

{
nbreHabitants = nbre;
}
}
	Pour changer le nombre d'habitant

private static int nbreInstances = 0;
public static int getNombreInstances()
{
return nbreInstancesBis;
}
	Variable commune, même valeur pour tous les objets
	Pour retourner le nombre d'instance

class Capitale extends Ville {
 	
private String monument;
 	
public Capitale(){
super();
 	
monument = "aucun";
}
 	
public Capitale(String nom, int hab, String pays, String monument){
super(nom, hab, pays);
 	
this.monument = monument;
}
 	
public String getAffiche(){
String str = super. getAffiche () + "\n \t ==>>" + this.monument+ " en est un monument";
 	
return str;
}
}
	Cree une classe Capitale, qui reprendre TOUS le code de la classe Ville
	
	Cree monument en privé
	
	Constructeur sans paramètre
	Ajoute les paramètres du constructeur sans paramètre de la classe mère
	Ajoute le paramètre monument
	
	Constructeur avec paramètre
	
	Ajoute les paramètres du constructeur avec paramètres de la classe mère
	Ajoute la paramètre monument avec la valeur de l'entree monument
	
	Modifie getAffiche lorsqu'il sera appeler
	Reprend la valeur par défaut et rajoute du texte
	
	Retourne str

Ville ville1 = new Ville("Marseille", 123456789, "France");	Cree la ville1 avec des paramètres
ville1.setNom("Marseille");	Change le nom de ville1 par marseille
System.out.print(ville1.getNom());	Affiche le nom de la ville

Lecture écriture fichier :
Vérifications :

File fichier = new File("test");	Cree l'objet file avec le nom de fichier
System.out.println("Chemin absolu du fichier : " + fichier.getAbsolutePath());	Donne le chemin absolu du fichier
System.out.println("Nom du fichier : " + fichier.getName());	Donne le nom du fichier
System.out.println("Est-ce qu'il existe ? " + fichier.exists());	Renvoie TRUE si le fichier existe
System.out.println("Est-ce un répertoire ? " + fichier.isDirectory());	Renvoie TRUE si le c'est un dossier
System.out.println("Est-ce un fichier ? " + fichier.isFile());	Renvoie TRUE si c'est un fichier
fichier.delete();	Supprime le fichier test (pour cette exemple)
Fichier.mkdir();	Cree un REPERTOIRE test (pour cette exemple)

Lecture ecriture:

String ligne;
BufferedReader fichier = new BufferedReader(new FileReader("test"));
while ((ligne = fichier.readLine()) != null) {
System.out.println(ligne);
}
	Cree variable ligne (contiendra une ligne)
	Ouvre le fichier test en mode lecture
	
	Temps qu'on est pas à la fin du fichier
	Affiche une ligne

BufferedWriter fichier = new BufferedWriter(new FileWriter("test"));
fichier.write("coucou");
fichier.newLine();
fichier.write("c\'est moi");
 	
fichier.close();
	Ouvre le fichier test en mode ecriture
	
	Ecrit "coucou" dans le fichier
	Met le curseur a la ligne suivante
	Ecrit "c'est moi" dans le fichier
	
	Ferme le fichier

NETBEANS

Cree un projet :

File / new project

Dans **categorie** : javaFX, dans **project** : javaFX application

[présentation de l'interfacegraphique ICI]

Dans le parti start() :

primaryStage.show();	Affiche la fenêtre vide **(dois toujours être a la fin du code)**
primaryStage.setTitle("Melordi");	Titre de la fenêtre
Group root = new Group(); Scene maScene = new Scene(root, 800, 600, Color.LIGHTBLUE); primaryStage.setScene(maScene);	Cree un nouveau groupe du nom root Cree une scène du nom maScene de talle 800*600 et de couleur bleu déclare la scène

Circle monCercle = new Circle();
monCercle.setCenterX(300);
monCercle.setCenterY(200);
monCercle.setRadius(100);
monCercle.setFill(Color.YELLOW);
monCercle.setStroke(Color.ORANGE);
monCercle.setStrokeWidth(5);
 	
root.getChildren().add(monCercle);
	Cree un objet cercle du nom monCercle
	Position X du cercle : 300
	Position Y du cercle : 200
	Rayon du cercle : 100
	Couleur du cercle : jaune
	Couleur du contour du cercle : orange
	Largeur du contour du cercle : 5
	
	affiche monCercle

Rectangle monRectangle = new Rectangle();
monRectangle.setX(300);
monRectangle.setY(200);
monRectangle.setWidth(300);
monRectangle.setHeight(200);
monRectangle.setFill(Color.GREEN);
monRectangle.setStroke(Color.DARKGREEN);
monRectangle.setStrokeWidth(5);
monRectangle.setArcHeight(30);
monRectangle.setArcWidth(30);
 	
root.getChildren().add(monRectangle);
	Cree un objet rectangle du nom monRectangle
	Position X du rectangle : 300
	Position Y du rectangle : 200
	Largeur du rectangle : 300
	Longueur du rectangle : 200
	Couleur du rectangle : vert
	Couleur du contour du rectangle : vert foncé
	Largeur du contour du rectangle : 5
	Arrondissement du haut et bas des coins : 30
	Arrondissement gauche et droite des coins : 30
	
	Affiche monRectangle

public class Clavier extends Parent{
 	
public Clavier(){
 	
}
}
	Crée une classe clavier qui hérite de la lasse Parent
	
	constructeur

Clavier monClavier = new Clavier(); root.getChildren().add(monClavier);	Cree un objet monCLavier Affiche monClavier

Audio avec des sons midi préenregistré:

public class Instru {	Classe Instru
public int volume = 100;	Variable volume en **public**
private Synthesizer synthetiseur; private MidiChannel canal;	Variable synthétiseur Variable canal
public Instru(){	Constructeur par défaut :

synthetiseur = MidiSystem.getSynthesizer();
synthetiseur.open();
canal = synthetiseur.getChannels()[0];
	Récupération du synthétiseur
	Ouverture du synthétiseur
	Récupération du canal

canal.programChange(0); }	Initialisation de l'instrument 0 **(ici le piano)**

public void note_on(int note){
canal.noteOn(note, volume);
}
	Joue la note dont le numéro est en paramètre

public void note_off(int note){
canal.noteOff(note);
}
	Arrête de jouer la note dont le numéro est en paramètre

public void set_instrument(int instru){
canal.programChange(instru);
}
}
	Change l'instrument dont le numéro est en paramètre

Base de données :

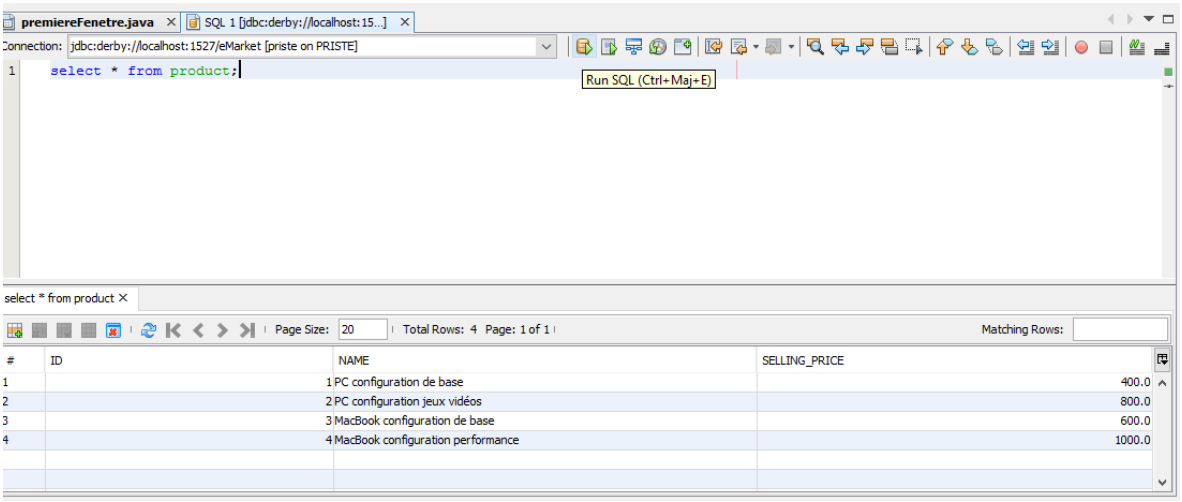
Clic droit libraries / add library / java DB driver

<u>Démarrage du serveur :</u>
Dans services / database / javaDB
clic droit sur javaDB / start serveur

<u>Création de la base de données eMark :</u>

Clic droit du javaDB / create database
mettre nom de la base, pseudo, mot de passe
Clique droit sur jbdc:derby://localhost:1527/nomBase puis execut command

Editeur SQL qui viens de s'ouvrir :
Fonctionne comme en SQL pour coder sur cette fenetre
run SQL pour lancer les commandes SQL



private static Connection connexion;	Cree une variable global connexion
connexion = DriverManager.getConnection("jdbc:derby://localhost:1527/eMarket", "priste", "priste63");	Connection à la base de donnée
Statement state = connexion.createStatement();	Cree un objet statement pour pouvoir communiquer avec la base
ResultSet result = state.executeQuery("SELECT * FROM membres");	Défini une requête en créant un objet ResultSet
ResultSetMetaData resultMeta = result.getMetaData();	Récupère les données de result
resultMeta.getColumnCount()	Nombre de colonne
resultMeta.getColumnName(1)	Nom de la 1ere colonne (ici id)
result.next();	Lit la ligne suivante
System.out.println(result.getObject(1));	Nom du premier objet de la ligne (ici 1)

result.close();
state.close();
connexion.close();
	Ferme result
	Ferme state
	Ferme connexion

Statement state = connexion.createStatement(); state.execute("REQUETE");	Pour autre requête qu'un SELECT

FONCTIONS UTILES

FONCTION	UTILISATION

```
public static String[][] CreationBase(String requete) {
    int nbLigne = 0;
    try {
        Statement state = connexion.createStatement();
        ResultSet result = state.executeQuery(requete);
        ResultSetMetaData resultMeta = result.getMetaData();

        while (result.next()) {
            nbLigne++;

        }
        state.close();
        result.close();

    } catch (SQLException ex) {
        Logger.getLogger(TestBDD.class.getName()).log(Level.SEVERE, null, ex);

    }

    try {
        Statement state = connexion.createStatement();
        ResultSet result = state.executeQuery(requete);
        ResultSetMetaData resultMeta = result.getMetaData();

        String laTable[][] = new String[resultMeta.getColumnCount() + 1][nbLigne + 1];
        for (int i = 1; i <= resultMeta.getColumnCount(); i++) {
            laTable[0][i] = resultMeta.getColumnName(1);
        }
        int i = 0;
        while (result.next()) {
            i++;
            for (int j = 1; j <= resultMeta.getColumnCount(); j++) {
                laTable[j][i] = String.valueOf(result.getObject(j));
            }

        }
        return laTable;
    } catch (SQLException ex) {
        Logger.getLogger(TestBDD.class.getName()).log(Level.SEVERE, null, ex);
        return null;
    }

}
```

Appel fonction :
String tableau [][] = CreationBase(requête);

Si tableau déjà créer:
Tableau = CreationBase(requête);

Utilization final:
Tableau[colonne][ligne]

Pourquoi ?

Permet de mettre toute la base dans un tableau a 2 dimension.

On retrouve ce qu'il y a dans la colonne 1 de la ligne 1 quand on fais [1][1].

[0][1] donne le nom de la première colonne

Appel d'une fenêtre Jframe à partir d'une autre :

Coding fen = new Coding(); fen.setVisible(true);	Coding : nom de la fenêtre Fen : nom de variable

Passer une variable d'un objet graphique a une Jframe :

clic droit sur l'objet – customize code
Mettre acces en public static
OK

Entre la creation variable et le set visible :

Coding.champValueFrame2.settext(this.champValueFrame1)	Le texte de champValueFrame2 sera toujours égale au texte de champValueFrame1

Pour envoyer a une variable de l'autre frame

public static String mail;	Dans la frame2 avant main
Coding.mail = this.champValueFrame1	Sur l'action de la frame 2 (toujours avant visible)