On souhaite écrire un programme qui réserve des voyages en train par une personne. Ci-dessous, un extrait du diagramme de classes vous propose certaines classes nécessaires pour la réservation. Rappel sur les niveaux de visibilité : **- private**, **+ public** et **# protected**

**reserver**

La classe **Voyage** fournit les informations suivantes :

* Le booléen *retour* indique s’il y a retour ou pas pour le voyage
* Le booléen *direct* indique si le voyage est un trajet direct ou pas
* L’entier *nb\_passagers* indique le nombre de passagers du voyage et l’entier *confort* donne la classe (1 ou 2)
* La méthode *aller(…)* copie la gare de départ, la gare d’arrivée, la date au format ″dd-MM-yyyy″ (exemple : ″15-05-2015″) et l’heure (entier) en paramètres dans les attributs pour l’aller du voyage
* La méthode *revenir(…)* copie la date et l’heure en paramètres dans les attributs pour le retour du voyage
* La méthode *reserver(…)* copie la personne qui réserve et le nombre de passagers en paramètres dans les attributs, puis instancie le tableau de passagers avec ce nombre de passagers.

La classe **Gare** fournit les informations suivantes :

* Le *nom* de la gare et un constructeur qui initialise son nom avec le paramètre

La classe **Personne** fournit les informations suivantes :

* Le *nom*, *prenom* et *email* de la personne, ainsi qu’un constructeur par défaut (sans paramètre)

**Question 1 (durée ~ 20 minutes. 5 points)**

1. [3 points]Complétez le diagramme de classes en respect des consignes suivantes (voir barème détaille +bas):

**Personne**

**# String** nom

**# String** prenom

**# String** email

1..\*

1

**Voyage**

* **boolean** retour
* **boolean** direct
* **int** nb\_passagers
* **int** confort
* **Date** date\_aller
* **Date** date\_retour
* **int** heure\_aller
* **int** heure\_retour

**Gare**

**- String** nom

1..\* **aller** 1

**+** Personne ()

**+** Gare (**String** nom)

1..\* **revenir** 1

\*

**stationner**

\*

**+ void** aller (…)

**+ void** revenir (…)

**+ void** reserver (…)

**Passager**

**Train**

**- String** tranche

**- String** carte

**- String** fidelite

**- int** numero

**- String** type

…

…

\*

1..\*

**Wagon**

**# int** numero

…

**WagonLit**

**WagonBar**

**WagonNormal**

**- int** nb\_lits

…

**- int** nb\_sieges

…

…

…

Barême détaillé :

Attributs de **Voyage** à compléter : **0,5 pt**.

**Voyageur**, ses attributs et relation d’héritage avec **Personne** : **0,5 pt**.

**Train**, ses attributs et relation d’association « stationner » et multiplicités avec **Gare** : **0,5 pt**.

**Wagon** et ses attributs : **0,5 pt**. Relation d’agrégation et multiplicités avec **Train** : **0,5 pt**.

**WagonNormal, WagonBar, WagonLit,** leurs attributs et relation d’héritage avec **Wagon** : **0,5 pt**.

* Dans la classe **Voyage** : les dates et les heures à l’aller et au retour.
* Un passager est une personne fournissant les informations suivantes : la tranche d’âge (exemples : ″26-59 ans″, ″60 et +″ etc.), la carte d’abonnement (exemples : ″carte 12-25″, ″famille nombreuse″ etc.) et le programme de fidélité (exemples : ″grand voyageur″, ″S’Miles″ etc.). La personne qui réserve n’est pas obligatoirement un passager mais un passager peut être la personne qui réserve.
* Des trains stationnent dans une gare.
* Un train est composé de wagons et fournit les informations suivantes : son numéro et son type (″TGV″, ″idTGV″ ou ″Corail″).
* Tout wagon a un numéro.
* Il y a 3 catégories de wagons : **WagonNormal** avec le nombre de sièges pour les passagers, **WagonLit** avec le nombre de lits et **WagonBar**.

*Vous indiquerez les nouvelles classes, les attributs avec leur niveau de visibilité et leur type, les relations inter-classes avec leurs éventuelles multiplicités et un verbe significatif pour chaque relation.*

1. [2 points]Répondre brièvement aux questions suivantes relatives au diagramme de classes :

* Quelles relations interclasses impliquent les 3 classes des « wagons » ? Doit-on nécessairement définir certaines de ces classes ? Justifier votre réponse.

Relation d’héritage. Le WagonBar pas nécessaire car aucun attribut ni méthode propre mentionnée. **(1 pt)**

* Les catégories de wagons rendent le wagon « général » inutilisable. Expliquez comment on modélise une classe qui existe mais ne sera jamais utilisée. Quel est l’intérêt d’une telle classe ?

Classe abstraite. Intérêt : factoriser les propriétés communes à tous les wagons : ici, le numéro. **(1 pt)**

**Question 2 (durée ~ 20 minutes. 5 points)**

*Pour chacune des classes suivantes, n’oubliez pas d’importer les éventuelles librairies nécessaires. Nous supposons que ces classes font partie du même package.*

Implémenter en Java la classe **Voyage** en respect desconsignes suivantes :

* Les attributs suivants en **private** :
* Les 4 attributs définis dans l’extrait du diagramme de la page précédente
* La date et l’heure à l’aller et au retour
* La personne qui réserve, un tableau de passagers et les gares de départ et d’arrivée

**(1 point)**

public class Voyage {

// Déclaration des attributs

private boolean retour, direct ;

private int nb\_passagers, confort ;

private java.util.Date date\_aller, date\_retour ; // String date accepté

private int heure\_aller, heure\_retour ;

private Gare depart, arrivee ;

private Personne connecte ;

private Passager[] voyageurs;

* Implémenter un premier constructeur par défaut (sans paramètre) qui initialise les 4 attributs définis dans l’extrait du diagramme selon les 4 règles suivantes : il n’y a pas de retour, le trajet est direct, il y a 1 seul passager et le confort est en classe 2.

**(1 point)**

public Voyage () {

this.retour = false ;

this.direct = true ;

this.nb\_passagers = 1 ;

this.confort = 2 ;

}

* Implémenter un second constructeur qui copie ses 4 paramètres respectivement dans les 4 attributs définis dans l’extrait du diagramme.

**(1 point)**

public Voyage (boolean retour, boolean direct, int nb, int classe) {

this.retour = retour ;

this.direct = direct ;

this.nb\_passagers = nb ;

this.confort = classe ;

}

* Implémenter les méthodes *aller(…)*, *revenir(…)* et *reserver(…)* en respect des informations indiquées dans la première page.

**(0,5 point)**

public void aller(Gare g, Gare a, java.util.Date d, int h) {

depart = g ; arrivee = a ; date\_aller = d ; heure\_aller = h ;

}

**(0,5 point)**

public void revenir(java.util.Date d, int h) {

date\_retour = d ; heure\_retour = h ;

}

**(1 point)**

public void reserver(Personne p, int nb) {

connecte = p; nb\_passagers = nb ;

voyageurs = new Passager[nb] ;

}

} // fin de la classe

**Question 3 (durée ~ 25 minutes. 5 points)**

Implémenter en Java une classe avec le **main** qui effectue les actions suivantes :

* Déclarer et lire au clavier (voir la classe **Scanner** en annexe page 4) les données suivantes sans tester leurs valeurs :
* Les booléens pour le retour et le trajet direct.
* Les noms des gares de départ et d’arrivée.
* La date à l’aller au format ″dd-MM-yyyy″ (voir classes **DateFormat** et **SimpleDateFormat** en annexe pages 4 et 5) et l’heure à l’aller.
* Le nombre de passagers et la classe de confort.
* Si le retour lu est vrai, lire la date et l’heure au retour.
* En cas d’erreur de format de lecture sur l’une des données, générer l’exception **Exception** et afficher un message d’erreur.

**(2,5 points)**

import java.text.SimpleDateFormat ;

import java.util.\*  ;

public class Controleur {

public static void main(String args[]) {

// Déclaration et initialisation des données

boolean retour, direct = true;

String date;

DateFormat dateStandard= new SimpleDateFormat("dd-MM-yyyy"); // formatage de date

Date date\_aller, date\_retour ;

int heure\_aller, heure\_retour, nb, classe ;

String nom\_depart, nom\_arrivee ;

System.out.println(″Lisez vos données (retour, date départ etc.) :″) ;

// Essai de lecture des données

try {

Scanner in = new Scanner(System.in) ;

retour = in.nextBoolean(); direct = in.nextBoolean();

nom\_depart = in.next(); nom\_arrivee= in.next();

date = in.next(); date\_aller = dateStandard.parse(date); // date fomatée

heure\_aller = in.nextInt();

nb = in.nextInt() ; classe = in.nextInt() ;

if (retour == true) {

System.out.println(″Lisez vos données de retour (date et heure) :″) ;

date = in.next();

date\_retour = dateStandard.parse(date); // date fomatée

heure\_retour = in.nextInt();

}

} // fin try

// Erreur de format : exception générée et sortie du programme

catch(Exception e) { System.out.println(“Erreur de format”); System.exit(1); }

* Instancier un objet de **Personne** avec son constructeur par défaut (sans paramètre).
* Instancier les gares de départ et d’arrivée (voir le constructeur de **Gare** dans le diagramme) avec les noms lus.
* Instancier un objet de **Voyage** avec votre second constructeur de la question précédente et les paramètres adéquats lus précédemment.

**(1,5 point)**

// Instancier les objets

Personne connecte = new Personne () ;

Gare depart = new Gare(nom\_depart) ;

Gare arrivee = new Gare(nom\_arrivee) ;

Voyage v = new Voyage(retour, direct, nb, classe) ;

* A partir de votre objet instancié de **Voyage**, appeler vos méthodes *aller(…),* *revenir(…)* seulement si le retour lu est vrai et *reserver(…)* de la question précédente avec les paramètres adéquats lus précédemment.

**(1 point)**

// Appel des méthodes avec les paramètres adéquats

v.aller(depart, arrivee, date\_aller, heure\_aller) ;

if (retour == true)

v.revenir(date\_retour, heure\_retour) ;

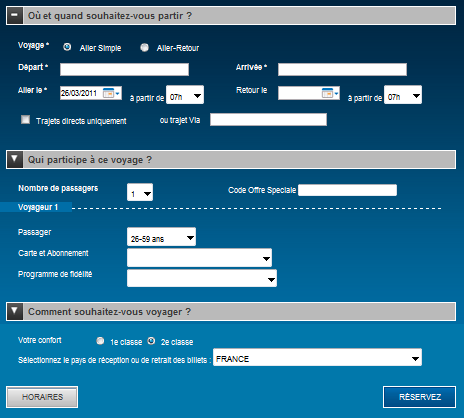
v.reserver(connecte, nb) ;

} // fin du main

} // fin de la classe

**Question 4 (durée ~ 25 minutes. 5 points)**

Votre dernière mission va être de mettre en œuvre une « vue graphique évènementielle » en vous inspirant du formulaire de réservation ci-dessous. Dernière ligne droite avant la fin du voyage ☺



**JRadioButton**

**JTextField**

**JCheckBox**

**JButton**

1. [1 point]Complétez les pointillés dans le formulaire en indiquant les noms des classes graphiques Swing

Voir en rouge ci-dessus : **0,25 point par nom de classe Swing**

1. [4 points]Implémenter en Java une classe pour cette « vue graphique évènementielle » qui hérite de **JFrame** (annexe page 6), implémente l’interface **ActionListener** (annexe page 6) en respect des consignes suivantes :

* Définir les attributs pour les objets graphiques Swing en pointillés ci-dessus (voir annexe pages 6 à 8).

**(1 point)**

import javax.swing.\* ;

import java.awt.\* ;

import java.awt.event.\* ;

public class Vue extends JFrame implements ActionListener {

// déclaration des attributs

private JButton reserve ;

private JRadioButton aller ;

private JCheckBox direct ;

private JTextField depart;

* Implémenter un constructeur par défaut (sans paramètre) qui instancie les objets Swing définis ci-dessus avec leur valeur par défaut.

**(1 point)**

public Vue() {

reserve = new JButton(″Réservez″) ;

aller = new JRadioButton(″Aller simple″, true) ;

direct = new JCheckBox(″Trajet direct uniquement″) ;

depart = new JTextField() ;

}

* Implémenter une méthode qui dessine la fenêtre du formulaire en lui donnant le titre, ses dimensions et la rend visible.

**(0,5 point)**

public void dessiner() {

super(″Où et quand souhaitez-vous partir ?″) ; // this.setTitle accepté aussi bien sûr

this.setSize(200, 100) ;

this.setVisible(true) ;

}

* Implémenter une méthode qui met en page (layout) la fenêtre du formulaire, y ajoute les objets Swing et met à l’écoute (listener) le bouton « Réservez ».

**(1 point)**

public void arranger() {

Container cp = this.getContentPane() ;

cp.setLayout(new GridLayout(3, 1)) ; // accepter tout layout coherent

cp.add(depart);

cp.add(arrivee);

cp.add(reserve);

reserve.addActionListener(this) ;

}

* Implémenter la méthode *actionPerformed(…)* qui ferme la fenêtre si l’objet cliqué est le bouton « Réservez ».

**(0,5 point)**

public void actionPerformed(ActionEvent e) {

if (e.getSource() == reserve)

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE) ; }

} // fin de la classe

**ANNEXE : extraits de la documentation API**

## java.util

## Class Scanner

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**Scanner**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#Scanner%28java.io.InputStream%29)([InputStream](http://docs.oracle.com/javase/1.5.0/docs/api/java/io/InputStream.html" \o "class in java.io) source)            Constructs a new Scanner that produces values scanned from the specified input stream. | |  |
| **Method Summary** | |
| void | [**close**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#close%28%29)()            Closes this scanner. |
| boolean | [**hasNext**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#hasNext%28%29)()            Returns true if this scanner has another token in its input. |
| [String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) | [**next**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#next%28%29)()            Finds and returns the next complete token from this scanner. |
| boolean | [**nextBoolean**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextBoolean%28%29)()            Scans the next token of the input into a boolean value and returns that value. |
| double | [**nextDouble**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextDouble%28%29)()            Scans the next token of the input as a double. |
| float | [**nextFloat**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextFloat%28%29)()            Scans the next token of the input as a float. |
| int | [**nextInt**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextInt%28%29)()            Scans the next token of the input as an int. |
| [String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) | [**nextLine**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#nextLine%28%29)()            Advances this scanner past the current line and returns the input that was skipped. |
| [Scanner](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html) | [**useDelimiter**](http://docs.oracle.com/javase/1.5.0/docs/api/java/util/Scanner.html#useDelimiter%28java.lang.String%29)([String](http://docs.oracle.com/javase/1.5.0/docs/api/java/lang/String.html) pattern)            Sets this scanner's delimiting pattern to a pattern constructed from the specified String. |

## 

## java.text

## Class DateFormat

[java.lang.Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html)

extended by[java.text.Format](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/Format.html)

extended by**java.text.DateFormat**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| protected | [**DateFormat**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html#DateFormat%28%29)()            Create a new date format. | |
| **Method Summary** | | |
| [**String**](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) | | [**format**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html#format%28java.util.Date%29)**(**[**Date**](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Date.html)**date)** Formats a Date into a date/time string. |
| [Date](http://docs.oracle.com/javase/1.4.2/docs/api/java/util/Date.html) | | [**parse**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html#parse%28java.lang.String%29)([String](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) source)            Parses text from the beginning of the given string to produce a date. |

## java.text

## Class SimpleDateFormat

[java.lang.Object](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/Object.html)

extended by[java.text.Format](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/Format.html)

extended by[java.text.DateFormat](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/DateFormat.html)

extended by**java.text.SimpleDateFormat**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**SimpleDateFormat**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#SimpleDateFormat%28java.lang.String%29)([String](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) pattern)            Constructs a SimpleDateFormat using the given pattern and the default date format symbols for the default locale. | |  |
| **Method Summary** | | |
| [**String**](http://docs.oracle.com/javase/1.4.2/docs/api/java/lang/String.html) | [**toPattern**](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#toPattern%28%29)()            Returns a pattern string describing this date format. | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Letter** | **Date or Time Component** | **Presentation** | **Examples** |  | **Date and Time Pattern** | **Result** |
| y | Year | [Year](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#year) | 1996; 96 |  | "dd-MM-yyyy" | 14-04-2012 |
| M | Month in year | [Month](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#month) | July; Jul; 07 |  |
| d | Day in month | [Number](http://docs.oracle.com/javase/1.4.2/docs/api/java/text/SimpleDateFormat.html#number) | 10 |  |  |  |

**java.awt**

**Class Container**

|  |  |
| --- | --- |
| **Method Summary** | |
| [Component](http://java.sun.com/javase/6/docs/api/java/awt/Component.html) | [**add**](http://java.sun.com/javase/6/docs/api/java/awt/Container.html#add%28java.awt.Component%29)([Component](http://java.sun.com/javase/6/docs/api/java/awt/Component.html) comp)            Appends the specified component to the end of this container. |
| void | [**setLayout**](http://docs.oracle.com/javase/6/docs/api/java/awt/Container.html#setLayout%28java.awt.LayoutManager%29)([LayoutManager](http://docs.oracle.com/javase/6/docs/api/java/awt/LayoutManager.html" \o "interface in java.awt) mgr)            Sets the layout manager for this container. |

## java.awt.event

## Interface ActionListener

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**actionPerformed**](http://java.sun.com/javase/6/docs/api/java/awt/event/ActionListener.html#actionPerformed%28java.awt.event.ActionEvent%29)([ActionEvent](http://java.sun.com/javase/6/docs/api/java/awt/event/ActionEvent.html" \o "class in java.awt.event) e)            Invoked when an action occurs. |

**java.util**

## Class EventObject

|  |  |  |
| --- | --- | --- |
| **Field Summary** | | |
| protected  [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html) | | [**source**](http://java.sun.com/javase/6/docs/api/java/util/EventObject.html#source)            The object on which the Event initially occurred. |
| **Method Summary** | | |
| [Object](http://java.sun.com/javase/6/docs/api/java/lang/Object.html) | [**getSource**](http://java.sun.com/javase/6/docs/api/java/util/EventObject.html#getSource%28%29)()            The object on which the Event initially occurred. | |

**javax.swing**

**Class JFrame**

|  |  |  |  |
| --- | --- | --- | --- |
| **Constructor Summary** | | | |
| [**JFrame**](http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html#JFrame%28%29)()            Constructs a new frame that is initially invisible. | | |  |
| [**JFrame**](http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html#JFrame%28java.lang.String%29)([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html) title)            Creates a new, initially invisible Frame with the specified title. | | |  |
| **Field Summary** | | | |
| static int | | [**EXIT\_ON\_CLOSE**](http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html#EXIT_ON_CLOSE)            The exit application default windows close operation. | |
| **Method Summary** | | | |
| [Container](http://java.sun.com/javase/6/docs/api/java/awt/Container.html) | [**getContentPane**](http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html#getContentPane%28%29)()            Returns the contentPane object for this frame. | | |
| void | [**setContentPane**](http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html#setContentPane%28java.awt.Container%29)([Container](http://java.sun.com/javase/6/docs/api/java/awt/Container.html) contentPane)            Sets the contentPane property. | | |
| void | [**setDefaultCloseOperation**](http://java.sun.com/javase/6/docs/api/javax/swing/JFrame.html#setDefaultCloseOperation%28int%29)(int operation)            Sets the operation that will happen by default when the user initiates a "close" on this frame. | | |
| void | [**setSize**](http://java.sun.com/javase/6/docs/api/java/awt/Window.html#setSize%28int,%20int%29)(int width, int height)            Resizes this component so that it has width and height. | | |
| void | [**setVisible**](http://java.sun.com/javase/6/docs/api/java/awt/Window.html#setVisible%28boolean%29)(boolean b)            Shows or hides this Window depending on the value of parameter b. | | |

## javax.swing

## Class AbstractButton

[java.lang.Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html)

extended by [java.awt.Component](http://docs.oracle.com/javase/6/docs/api/java/awt/Component.html)

extended by [java.awt.Container](http://docs.oracle.com/javase/6/docs/api/java/awt/Container.html)

extended by [javax.swing.JComponent](http://docs.oracle.com/javase/6/docs/api/javax/swing/JComponent.html)

extended by **javax.swing.AbstractButton**

|  |  |
| --- | --- |
| **Method Summary** | |
| void | [**addActionListener**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#addActionListener%28java.awt.event.ActionListener%29)([ActionListener](http://docs.oracle.com/javase/6/docs/api/java/awt/event/ActionListener.html" \o "interface in java.awt.event) l)            Adds an ActionListener to the button. |
| [String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) | [**getText**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#getText%28%29)()            Returns the button's text. |
| void | [**setText**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#setText%28java.lang.String%29)([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) text)           Sets the button's text. |

## javax.swing

## Class JRadioButton

[java.lang.Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html)

extended by [java.awt.Component](http://docs.oracle.com/javase/6/docs/api/java/awt/Component.html)

extended by [java.awt.Container](http://docs.oracle.com/javase/6/docs/api/java/awt/Container.html)

extended by [javax.swing.JComponent](http://docs.oracle.com/javase/6/docs/api/javax/swing/JComponent.html)

extended by [javax.swing.AbstractButton](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html)

extended by [javax.swing.JToggleButton](http://docs.oracle.com/javase/6/docs/api/javax/swing/JToggleButton.html)

extended by **javax.swing.JRadioButton**

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**JRadioButton**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JRadioButton.html#JRadioButton%28%29)()            Creates an initially unselected radio button with no set text. |  |
| [**JRadioButton**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JRadioButton.html#JRadioButton%28java.lang.String,%20boolean%29)([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) text, boolean selected)            Creates a radio button with the specified text and selection state. |  |

**javax.swing**

**Class JButton**

[java.lang.Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html)

extended by [java.awt.Component](http://docs.oracle.com/javase/6/docs/api/java/awt/Component.html)

extended by [java.awt.Container](http://docs.oracle.com/javase/6/docs/api/java/awt/Container.html)

extended by [javax.swing.JComponent](http://docs.oracle.com/javase/6/docs/api/javax/swing/JComponent.html)

extended by [javax.swing.AbstractButton](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html)

extended by **javax.swing.JButton**

|  |  |
| --- | --- |
| **Constructor Summary** | |
| [**JButton**](http://java.sun.com/javase/6/docs/api/javax/swing/JButton.html#JButton%28%29)()           Creates a button with no set text or icon. |  |
| [**JButton**](http://java.sun.com/javase/6/docs/api/javax/swing/JButton.html#JButton%28java.lang.String%29)([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html) text)           Creates a button with text. |  |

## javax.swing

## Class JTextField

[java.lang.Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html)

extended by [java.awt.Component](http://docs.oracle.com/javase/6/docs/api/java/awt/Component.html)

extended by [java.awt.Container](http://docs.oracle.com/javase/6/docs/api/java/awt/Container.html)

extended by [javax.swing.JComponent](http://docs.oracle.com/javase/6/docs/api/javax/swing/JComponent.html)

extended by [javax.swing.text.JTextComponent](http://docs.oracle.com/javase/6/docs/api/javax/swing/text/JTextComponent.html)

extended by **javax.swing.JTextField**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**JTextField**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JTextField.html#JTextField%28%29)()            Constructs a new TextField. | |  |
| [**JTextField**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JTextField.html#JTextField%28java.lang.String%29)([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) text)            Constructs a new TextField initialized with the specified text. | |  |
| **Method Summary** | | |
| [String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) | [**getText**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#getText%28%29)()            Returns the button's text. | |
| void | [**setText**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#setText%28java.lang.String%29)([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) text)           Sets the button's text. | |
| void | [**addActionListener**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#addActionListener%28java.awt.event.ActionListener%29)([ActionListener](http://docs.oracle.com/javase/6/docs/api/java/awt/event/ActionListener.html" \o "interface in java.awt.event) l)            Adds an ActionListener to the button. | |

## 

## javax.swing

## Class JCheckBox

[java.lang.Object](http://docs.oracle.com/javase/6/docs/api/java/lang/Object.html)

extended by [java.awt.Component](http://docs.oracle.com/javase/6/docs/api/java/awt/Component.html)

extended by [java.awt.Container](http://docs.oracle.com/javase/6/docs/api/java/awt/Container.html)

extended by [javax.swing.JComponent](http://docs.oracle.com/javase/6/docs/api/javax/swing/JComponent.html)

extended by [javax.swing.AbstractButton](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html)

extended by [javax.swing.JToggleButton](http://docs.oracle.com/javase/6/docs/api/javax/swing/JToggleButton.html)

extended by **javax.swing.JCheckBox**

|  |  |  |
| --- | --- | --- |
| **Constructor Summary** | | |
| [**JCheckBox**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JCheckBox.html#JCheckBox%28%29)()            Creates an initially unselected check box button with no text, no icon. | |  | |
| [**JCheckBox**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JCheckBox.html#JCheckBox%28java.lang.String%29)([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) text)            Creates an initially unselected check box with text. | |  | |
| [**JCheckBox**](http://docs.oracle.com/javase/6/docs/api/javax/swing/JCheckBox.html#JCheckBox%28java.lang.String,%20boolean%29)([String](http://docs.oracle.com/javase/6/docs/api/java/lang/String.html) text, boolean selected)            Creates a check box with text and specifies whether or not it is initially selected. | |  | |
| **Method Summary** | | |
| boolean | [**isSelected**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#isSelected%28%29)()            Returns the state of the button.. | |
| void | [**setSelected**](http://docs.oracle.com/javase/6/docs/api/javax/swing/AbstractButton.html#setSelected%28boolean%29)(boolean b)            Sets the state of the button. | |