Parallele Programmierung

Java Schlüsselwörter
synchronized (angewendet auf Methodenname)
synchronized(object) { .. }

Klasse Object
public void wait() throws InterruptedException()
public void wait(long millis) throws InterruptedException()
public void notify()
public void notifyAll()

Schnittstelle Runnable
public void run()

Klasse Thread
public Thread ()
public Thread(String name)
public Thread(Runnable r)
public Thread(Runnable r, String name)
public static Thread currentThread()
public String getName()
public void interrupt()
public boolean isAlive()
public boolean isInterrupted()
public void join() throws InterruptedException()
public void join(long millis) throws InterruptedException()
public void run()
public void setName(String name)
public void start() throws IllegalThreadStateException
public static void sleep(long millis) throws InterruptedException()

Klasse Semaphore
public Semaphore(int init)
public void p()
public void v()

Klasse AdditiveSemaphore
public AdditiveSemaphore (int init)
public void p()
public void p(int x)
public void v()
public void v(int x)

Klasse SemaphoreGroup
public SemaphorGroup(int numberOfMembers)
public void changeValues(int[] deltas)

Schnittstelle Lock
public void lock()
public void unlock()
public Condition newCondition()

Schnittstelle Condition
public void await() throws InterruptedException
public void awaitUninterruptibly()
public void signal()
public void signalAll()

Klasse ReentrantLock implements Lock
parameterloser Konstruktor

Graphische Benutzeroberflächen

Grundstruktur einer JavaFX-Anwendung
… extends Application
Überschreiben von public void start(Stage stage):
    Aufbau der Oberfläche, Scene erzeugen mit Wurzel, stage.setScene(scene),
    stage.setTitle(title), stage.show()
main mit launch(args)

Schnittstelle ChangeListener<T>:
public void changed(ObservableValue<? extends T> observable, T oldValue, T newValue)

Properties
SimpleBooleanProperty, SimpleStringProperty, SimpleIntegerProperty, SimpleDoubleProperty
Konstruktoren: parameterlos
Methoden getValue und setValue
public void addListener(ChangeListener<…> listener)
Binding unidirektional (ohne Berechnung von Werten): public void bind(ObsevableValue<…> obs)
Binding bidirektional: public void bindBidirectional(Property<…> other)

FXCollections
Erzeugung von ObservableList über FXCollections:
    ObservableList<…> observableList = FXCollections.observableList(simpleList)
Hinzufügen: observableList.addAll(element1, element2, element3)
Entfernen: observableList.removeAll(element1, element2, element3)

Container
Pane, HBox, VBox, FlowPane, BorderPane, GridPane
Konstruktoren ohne Argumente
Elemente hinzufügen:
    über getChildren() (liefert ObservableList)
    aber mit Spezialmethoden für BorderPane (setTop, setLeft, setBottom, setRight, setCenter)
    und mit Spezialmethoden für GridPane (add(element, columnIndex, rowIndex) oder
    add(element, columnIndex, rowIndex, columnSpan, rowSpan)

Klasse Label
public Label(String text)
public String getText()
public void setText(String text)
public StringProperty textProperty()

Schnittstelle EventHandler<T extends Event>
public void handle(T event)

Klasse ActionEvent

Klasse Button
public Button(String text)
public String getText()
public void setText(String text)
public StringProperty textProperty()
public void setOnAction(EventHandler<ActionEvent> action)

Klasse ToggleGroup
parameterloser Konstruktor

Klasse RadioButton
public RadioButton(String text)
public void setText(String text)
public void setOnAction(EventHandler<ActionEvent> action)
public void setSelected(boolean b)
public boolean isSelected
public BooleanProperty selectedProperty()
mehrere RadioButtons zu ToggleGroup zusammenfassen:

toggleGroup.getToggles().addAll(radioButton1, radioButton2, radioButton3)
oder für jeden RadioButton rb: rb.setToggleGroup(toggleGroup)

Klasse CheckBox
public CheckBox(String text)
public void setText(String text)
public void setOnAction(EventHandler<ActionEvent> action)
public void setSelected(boolean b)
public boolean isSelected
public BooleanProperty selectedProperty()

Klasse ComboBox<T>
public ComboBox()
public ComboBox(ObservableList<T> items)
public void setPromptText(String text)
public void setEditable(boolean value)
public void setOnAction(EventHandler<ActionEvent> action)
public ObjectProperty<T> valueProperty()

Klasse Slider
public Slider(double min, double max, double value)
public void setValue(double n)
public double getValue()
public void setOrientation(Orientation orient)

mit Orientation.HORIZONTAL oder Orientation.VERTICAL

public DoubleProperty valueProperty()

Klasse ListView<T>
public ListView()

public ListView(ObservableList<T> items)
Hinzufügen und Entfernen: list.getItems() liefert ObservableList
selektierte Elemente abfragen:
    ObservableList<T> sels = listView.getSelectionModel().getSelectedItems()
auf Selektionsereignis reagieren:
    listView.getSelectionModel().selectedItemProperty().addListener(changeListener)

Klasse TextField
public TextField()
public TextField(String text)
public void setText(String text)
public String getText()
public void setPromptText(String text)
public void setOnAction(EventHandler<ActionEvent> action)
public StringProperty textProperty()

Klasse PasswordField
public PasswordField()
public void setText(String text)
public String getText()
public void setPromptText(String text)
public void setOnAction(EventHandler<ActionEvent> action)
public StringProperty textProperty()

Klasse TextArea
public TextArea()
public TextArea(String text)
public void setText(String text)
public String getText()
public void setPromptText(String text)
public StringProperty textProperty()

Klasse Stage
parameterloser Konstruktor

Klasse MouseEvent
public void double getX() / public void double getY():
    Mausposition relativ zur Quelle des Ereignisses,
    hängt davon, an welchem Objekt Listener angemeldet wurde

Grafik
als Container Pane verwenden
Klasse Line: Konstruktor public Line(double startX, double startY, double endX, double endY)
Klasse Rectangle: Konstruktor public Rectangle(double x, double y, double width, double height)
Klasse Circle: Konstruktor public Circle(double centerX, double centerY, double radius)
Farbe der Begrenzungslinie: element.setStroke(Color…) wie z.B. Color.RED oder Color.BLUE
Dicke der Linie: element.setStrokeWidth(1) oder element.setStrokeWidth(5)
für Rectangle und Circle:
    ausgefüllt: element.setFill(Color.BLACK)
    nicht ausgefüllt: element.setFill(null)
Reaktion auf Mausereignisse: Anmelden von Listenern an Pane oder Grafikelementen:
    setOnMousePressed, setOnMouseReleased, setOnMouseMoved, setOnMouseDragged

Parameter von setOnMouse*: EventHandler<MouseEvent>

Klasse Platform
public static void runLater(Runnable doRun)

Sockets

Klasse InetAdress
public static InetAddress getByName(String host)   throws UnknownHostException
public static InetAddress[] getAllByName(String host) throws UnknownHostException

Klasse DatagramPacket
public DatagramPacket(byte buffer[], int length)
public DatagramPacket(byte buffer[], int length, InetAddress address, int port)
public byte[] getData()
public void setData(byte[] buffer)
public int getLength()
public void setLength(int length)
public InetAddress getAddress()
public void setAddress(InetAddress address)
public int getPort()
public void setPort(int port)

Klasse DatagramSocket
public DatagramSocket() throws SocketException()
public DatagramSocket(int port) throws SocketException()
public void send(DatagramPacket p) throws IOException
public void receive(DatagramPacket p) throws IOException
public void setSoTimeout(int timeout) throws SocketException
public void close()

Klasse MulticastSocket
public MulticastSocket() throws IOException
public MulticastSocket(int port) throws IOException
public void joinGroup(InetAddress mcastaddress) throws IOException
public void leaveGroup(InetAddress mcastaddress) throws IOException

Klasse ServerSocket
public ServerSocket(int port) throws IOException
public Socket accept() throws IOException
public void close() throws IOException

Klasse Socket
public Socket(String host, int port) throws UnknownHostException, IOException
public InputStream getInputStream() throws IOException
public OutputStream getOutputStream() throws IOException
public void close() throws IOException

Klasse OutputstreamWriter
public OutputstreamWriter(OutputStream os)

Klasse BufferedWriter
public BufferedWriter(Writer out)
public void write(String text) throws IOException
public void newLine() throws IOException

Klasse InputStreamReader
public InputStreamReader(InputStream is)

Klasse BufferedReader
public BufferedReader(Reader in)
public String readLine() throws IOException

Klasse DataOutputStream
public DataOutputStream(OutputStream out)
public void writeBoolean(boolean b) throws IOException
public void writeInt (int i) throws IOException
public void writeLong(long l) throws IOException
public void writeDouble(double d) throws IOException

Klasse DataInputStream
public DataInputStream(InputStream in)
public boolean readBoolean() throws IOException
public int readInt() throws IOException
public long readLong() throws IOException
public double readDouble() throws IOException

RMI

Klasse UnicastRemoteObject
public UnicastRemoteObject() throws RemoteException

Schnittstelle Serializable

Ausnahme RemoteException

Klasse Naming
public static void rebind(String name, Remote obj)
    throws MalformedURLException, RemoteException
public static Remote lookup(String name)
    throws NotBoundException, MalformedURLException, RemoteException

Servlets / JSF

HTML
<form method=“get|post“ action=“testaction“>
<input type=“text“ name=“bla“ size=“40“>
<input type=“password“ name=“bla“ size=“40“>
<textarea name="user_eingabe" cols="50" rows="10"></textarea>
<input type=“checkbox“ name=“Blub“>
<input type=“radio“ name=“bla“ value=“testvalue“>

<input type=“submit“ value=“Testbutton“>

Klasse HttpServlet
public void doGet(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException)
public void doPost(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException)
public ServletContext getServletContext()

Klasse HttpServletRequest
public String getParameter(String name)
public String[] getParameterValues(String name)
public Cookie[] getCookies()
public HttpSession getSession(boolean create)
public Object getAttribute(String name)
public void setAttribute(String name, Object value)

Klasse HttpServletResponse
public PrintWriter getWriter() throws IOException
public void setStatus(int statusCode)
public void addCookie(Cookie cookie)

Klasse ServletContext
public Object getAttribute(String name)
public void setAttribute(String name, Object value)
public void removeAttribute(String name)

Klasse HttpSession
public Object getAttribute(String name)
public void setAttribute(String name, Object value)
public void removeAttribute(String name)
public void invalidate()

JSF
Annotation @ManagedBean
Annotationen @ApplicationScoped, @SessionScoped, @ViewScoped und @RequestScoped
XHTML-Formular:

```
    <h:form>
        <h:inputText value=“#{…}“ />
        <h:outputText value=“#{…}“ />
        <h:commandButton value=“…“ action=”…” />
    </h:form>
```