Heute

- Grafische Benutzeroberflächen
- (Anonyme) innere Klassen
- Einfache Simulationen
- main / ausführbare jar-Files

- Game of Life

Organisatorisches

- Heute letzte UE-Einheit
- VO am Di 21.1., 9:30 (**CR IL/IT**):
  Fragestunde
- VO am 30.1.? (bei Interesse Rekursion)

- UE-Abschlusstest am 23.1., 11h (HR)
- VO-Prüfung am 22.1., 10 Uhr (HS Miller)
- Anmeldung zur VO-Prüfung!

- VO wird evaluiert. Nehmen Sie teil!

Grafische Oberflächen (GUI)

- Pakete *awt* (alt) und *swing* (neu) enthalten Klassen für GUIs.
- *swing* verwendet einige Klassen aus *awt*, sodass meist beide importiert werden müssen.
- Klassennamen in *swing*, die Klassen aus *awt* ersetzen, beginnen typischerweise mit einem 'J' (z.B. *JFrame* ersetzt *Frame*).

GUI: Wichtige Klassen und Methoden

- *JFrame* ist Klasse für Fenster.

  Konstruktor erzeugt leeres, unsichtbares Fenster.
- Auf Inhalt des Fensters wird mit

  Container getContentPane()

  zugegriffen.
- Mit *setVisible(boolean)* kann Fenster (un)sichtbar gemacht werden.

GUI: Wichtige Klassen und Methoden

- Klassen für Steuerelemente u.ä.: *JButton*, *JLabel*, *JTextField*, *JmenuBar*, ...
- Steuerelemente können dem Inhalt (Objekt der Klasse *Container*) des Fensters mit *add* hinzugefügt werden.
- Damit mehrere Elemente sichtbar, muss mit *setLayout()* die Anordnung festgelegt werden.

GUI: Benutzinteraktion

- Für jedes Steuerelement gibt es ein oder mehrere “Listener”-Interfaces, die Methoden vorgeben, die ausgeführt werden, wenn der Benutzer bestimmte Aktionen durchführt (Mausklick, Eingabe, o.ä.).
- Die Steuerelemente müssen mit den entsprechenden “Listeners” verknüpft werden.

GUI: Benutzinteraktion

Beispiel:

JButton knopf=new JButton("drk");

- Damit Button reagiert, implementiere Interface

  ActionListener

  und schreibe Methode

  void actionPerformed(Event e)

- Button mit Listener verknüpfen:

  knopf.addActionListener(this);

Innere Klassen

- Um Fallunterscheidungen bei Benutzerinteraktion zu vermeiden (welches Steuerelement hat Ereignis ausgelöst?), können *innere Klassen* verwendet werden.
- Diese Klassen sind nur innerhalb der umschließenden Klasse bekannt, können aber auf Attribute der umschließenden Klasse zugreifen.

Innere Klassen

- Beispiel:

```
class EinFenster{
    JButton knopf = new JButton(...);
    knopf.addActionListener(new Innen());

    // innere Klasse
    class Innen implements ActionListener
    {
        public void actionPerformed(...)
        {...}
     }
}
```

Anonyme innere Klassen

- Da von inneren Klassen meist nur ein Objekt erzeugt wird, gibt es die Möglichkeit, innere Klassen anonym zu machen.
- In diesem Fall wird an Ort und Stelle der (anonyme) Konstruktor des zu erzeugenden Objekts aufgerufen und die entsprechende Methode implementiert.

Anonyme innere Klassen

- Beispiel:

```
class EinFenster{
  JButton knopf = new JButton(...);
  knopf.addActionListener(
      // Konstruktoraufruf der anonymen
      // inneren Klasse
      new ActionListener(){
            public void actionPerformed(...)
              {...}
      });
}
```

public static void main(String[] args)

- Definiert die „Hauptmethode" einer Klasse.
- Diese Methode wird ausgeführt, wenn das Java-Laufzeitsystem mit dieser Klasse als Input gestartet wird.
- Z.B. *java TestKlasse*
- Dazu muss *TestKlasse.class* im entsprechenden Verzeichnis vorhanden sein, und in Testklasse eine *main*-Methode definiert sein.
- Alle weiteren benötigten Klassen müssen ebenfalls als *class*-Files vorhanden sein.
- Im String-Array *args* werden die beim Aufruf eingegebenen Argumente zur Verfügung gestellt.

Ausführbares *jar*-File

- In BlueJ kann ausführbares *jar*-File erstellt werden.
- Dazu muss Klasse angegeben werden, die *main*-Methode enthält.
- *jar*-File kann durch Doppelklick ausgeführt werden, wenn JDK und JRE auf Rechner installiert und mit *jar*-Files verknüpft.