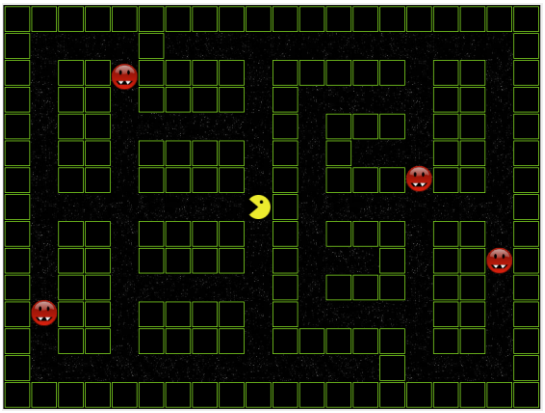
OOP: PacBoy

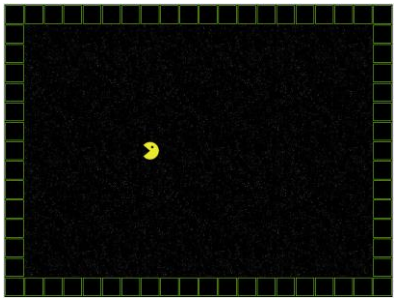
Das Spiel, das wir programmieren werden, orientiert sich am legendären Spiel Pac-Man aus dem Jahr 1980. Ein gelber Smiley bewegt sich durch ein Labyrinth und sammelt Früchte auf. Dabei darf der Smiley nicht mit einem Geist zusammenstoßen.



PacBoy steuern, Bild auswählen, fillWorld()
(→ PacBoy_00)

```
public class PacBoy  extends Actor
{
   public void act()
   {
      // Tasten prüfen:
      this.checkKeys();
   }

   public void checkKeys()
   {
      // Objekt in die gewünschte Richtung bewegen und passendes Bild anzeigen:
      if(Greenfoot.isKeyDown("left"))
      {
         this.setLocation(getX()-2, getY());
         this.setImage("pacboy_40px_links.png");
      }

      if(Greenfoot.isKeyDown("right"))
      {
         this.setLocation(getX()+2, getY());
         this.setImage("pacboy_40px_rechts.png");
      }

      if(Greenfoot.isKeyDown("up"))
      {

         this.setLocation(getX(), getY()-2);
         this.setImage("pacboy_40px_oben.png");
      }

      if(Greenfoot.isKeyDown("down"))
      {
         this.setLocation(getX(), getY()+2);
         this.setImage("pacboy_40px_unten.png");
      }
   }
}

public class PacWorld  extends World
{
   public PacWorld()
   {
      // Create a new world with 800x600 cells with a cell size of 1x1 pixels.
      super(800, 600, 1);
      fillWorld();
   }

   public void fillWorld()
   {
      PacBoy meinPacBoy;
      meinPacBoy = new PacBoy();
      this.addObject(meinPacBoy, 300, 300);
   }
```

```
}
```

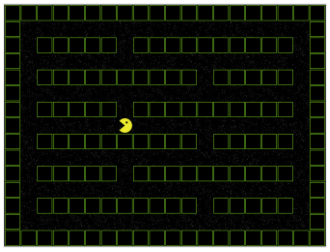


Wände oben, unten, links, rechts
(→ PacBoy_01)

```
public class PacWorld  extends World
{
   public PacWorld()
   {
      // Create a new world with 800x600 cells with a cell size of 1x1 pixels.
      super(800, 600, 1);
      fillWorld();
   }

   public void fillWorld()
   {
      PacBoy meinPacBoy;
      meinPacBoy = new PacBoy();
      this.addObject(meinPacBoy, 300, 300);

      // Wall-Objekte ganz oben mit Zähler-Schleife einfügen!
      // Beispiel für Zählerschleife, die etwas 10 mal ausführt,
      // indem sie von 0 bis 9 zählt:
      //       for(zaehler = 0; zaehler <= 9; zahler = zaehler + 1)
      //       {
      //       }
      int zaehler;

      for(zaehler = 20; zaehler <= 780; zaehler = zaehler + 40)
      {
         this.addObject(new Wall(), zaehler, 20);
      }

      // Wall-Objekte ganz unten:
      for(zaehler = 20; zaehler <= 780; zaehler = zaehler + 40)
      {
         this.addObject(new Wall(), zaehler , 580);
      }

      // Wall-Objekte ganz links:
      for(zaehler = 20; zaehler <= 580; zaehler = zaehler + 40)
      {
         this.addObject(new Wall(), 20, zaehler);
      }

      // Wall-Objekte ganz rechts:
      for(zaehler = 20; zaehler <= 580; zaehler = zaehler + 40)
      {
         this.addObject(new Wall(), 780, zaehler);
      }
   }
}
```

Mehr Wände
(→ PacBoy_02)

Wir ergänzen die fillWorld-Methode um den folgenden Code, um mehr Wände einzufügen:



```
   public void fillWorld()
   {
         ……
      // ########################  Mehr Wände ####################

      // 1. Reihe von links nach rechts
      for(zaehler = 100; zaehler <= 700; zaehler = zaehler + 40)
      {
         if(zaehler != 300)
         {
```

```
            this.addObject(new Wall(), zaehler, 100);
        }
    }

    // 2. Reihe von links nach rechts
    for(zaehler = 100; zaehler <= 700; zaehler = zaehler + 40)
    {
        if(zaehler != 500)
        {
            this.addObject(new Wall(), zaehler , 180);
        }
    }

    // 3. Reihe von links nach rechts
    for(zaehler = 100; zaehler <= 700; zaehler = zaehler + 40)
    {
        if(zaehler != 300)
        {
            this.addObject(new Wall(), zaehler , 260);
        }
    }

    for(zaehler = 100; zaehler <= 700; zaehler = zaehler + 40)
    {
        if(zaehler != 500)
        {
            this.addObject(new Wall(), zaehler , 340);
        }
    }

    for(zaehler = 100; zaehler <= 700; zaehler = zaehler + 40)
    {
        if(zaehler!= 300)
        {
            this.addObject(new Wall(), zaehler , 420);
        }
    }

    for(zaehler = 100; zaehler <= 700; zaehler = zaehler + 40)
    {
        if(zaehler != 500)
        {
            this.addObject(new Wall(), zaehler , 500);
        }
    }
}
```

Kollision mit Wänden
(→ PacBoy_03)

Der PacBoy kann sich bisher noch durch die Wände bewegen. Das soll jetzt geändert werden.
Lösungsidee: Wir bewegen uns in die gewünschte Richtung. Danach prüfen wir, ob es eine Kollision mit der Wand gab. Fall ja, bewegen wir den PacBoy wieder zurück (Das geht so schnell, dass man es nicht sehen kann).

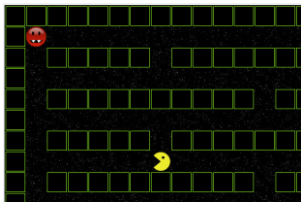
```
public class PacBoy  extends Actor
{
    public void act()
    {
        // Tasten prüfen:
        this.checkKeys();
    }

    public void checkKeys()
    {
        // Objekt in die gewünschte Richtung bewegen und passendes Bild anzeigen:
        if(Greenfoot.isKeyDown("left"))
        {
            // Objekt in die gewünschte Richtung bewegen:
            this.setLocation(getX()-2, getY());
```

```
        // Wenn KEINE Kollision mit Mauer => Bild ändern, ansonsten: Objekt zurück bewegen
        if(this.checkForWall() == false)
        {
            this.setImage("pacboy_40px_links.png");
        }
        else
        {
            this.setLocation(getX()+2, getY());
        }
    }

    if(Greenfoot.isKeyDown("right"))
    {
        this.setLocation(getX()+2, getY());

        if(this.checkForWall() == false)
        {
            this.setImage("pacboy_40px_rechts.png");
        }
        else
        {
            this.setLocation(getX()-2, getY());
        }
    }

    if(Greenfoot.isKeyDown("up"))
    {
        this.setLocation(getX(), getY()-2);

        if(this.checkForWall() == false)
        {
            this.setImage("pacboy_40px_oben.png");
        }
        else
        {
            this.setLocation(getX(), getY()+2);
        }
    }

    if(Greenfoot.isKeyDown("down"))
    {
        this.setLocation(getX(), getY()+2);

        if(this.checkForWall() == false)
        {
            this.setImage("pacboy_40px_unten.png");
        }
        else
        {
            this.setLocation(getX(), getY()-2);
        }
    }
}

// Wir überprüfen, ob das PacBoy-Objekt mit einer Mauer zusammenstößt in der Methode checkForWall().
// Diese Methode ist nicht void, weil Sie etwas zurückgibt, nämlich 'true' oder 'false'
// Methoden, die nur einen der beiden Werte zurückgeben können, haben den Rückgabetyp 'boolean'.
public boolean checkForWall()
{
    boolean kollision;
    kollision = false;

    Actor wall = getOneIntersectingObject( Wall.class );
    if(wall!=null)
    {
        kollision = true;
    }
    return kollision;
}
```

}

Geister bewegen
(→ PacBoy_04)

Wir erstellen die Klasse Ghost*. Dann ergänzen wir die fillWorld-Methode so, dass ein Ghost-Objekt eingefügt wird.*



```
public class PacWorld  extends World
{
   public PacWorld()
   {
      // Create a new world with 800x600 cells with a cell size of 1x1 pixels.
      super(800, 600, 1);
      fillWorld();
   }

   public void fillWorld()
   {
      …….

      Ghost geist1;
      geist1 = new Ghost();
      this.addObject(geist1, 60, 60);
   }
}
```

Jetzt sorgen wir dafür, dass sich das Ghost-Objekt automatisch von links nach rechts bewegt und anhält, wenn es mit einer Wand zusammenstößt.

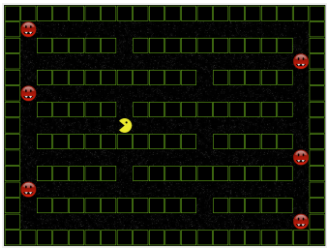
```
public class Ghost extends Actor
{
   public void act()
   {
      this.setLocation(this.getX() + 2, this.getY());

      if(this.checkForWall() == true)
      {
         this.setLocation(this.getX() - 2, this.getY());
      }
   }

   public boolean checkForWall()
   {
      boolean kollision;
      kollision = false;

      Actor wall = getOneIntersectingObject( Wall.class );
      if(wall!=null)
      {
         kollision = true;
      }
      return kollision;
   }
}
```

Das funktioniert. Allerdings wollen wir, dass das Ghost-Objekt umkehrt, also die Richtung ändert, wenn es an eine Wand stößt. Das funktioniert mit unserer bisherigen Lösung aber nicht (oder nur sehr umständlich).
*Lösung: Wir schreiben nicht direkt in den Code, wieviele Pixel sich das Objekt bewegen soll. Statt dessen verwenden wir eine Variable (*xRichtung*), in der wir diesen Wert speichern. Wenn wir dann an eine Wand stoßen, kehren wir diesen Wert um. Unser Programmcode muss so verändert werden:*

```
public class Ghost extends Actor
{
   private int xRichtung = 2;

   public void act()
   {
```

```
        this.setLocation(this.getX() + xRichtung, this.getY());

        if(this.checkForWall() == true)
        {
            this.setLocation(this.getX() - xRichtung, this.getY());
            xRichtung = - xRichtung;
        }
    }

    public boolean checkForWall()
    {
        boolean kollision;
        kollision = false;

        Actor wall = getOneIntersectingObject( Wall.class );
        if(wall!=null)
        {
            kollision = true;
        }
        return kollision;
    }
}
```

Fügen Sie jetzt noch fünf weitere Geister ein!



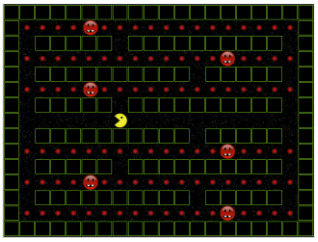
Kollision mit Ghost
(→ PacBoy_05)

Der PacBoy erhält eine CheckCollision-Methode und prüft, ob er mit einem Ghost-Objekt zusammengestoßen ist.

```
public class PacBoy  extends Actor
{
    public void act()
    {
        // Tasten prüfen:
        this.checkKeys();
        this.checkCollision();
    }


    public void checkCollision()
    {
      Actor einGeist;
      einGeist = this.getOneIntersectingObject(Ghost.class);

      if(einGeist != null)
      {
        this.getWorld().addObject(new Skull(), this.getX(), this.getY()) ;
        this.getWorld().removeObject(this);
      }
    }
}
```

Food-Objekte einfügen
(→ PacBoy_06)

Wir fügen mit Schleifen Food-Objekte in die Welt ein, die der PacBoy dann im nächsten Schritt aufsammeln kann. Diese Food-Objekte müssen in der fillWorld-Methode als erstes eingefügt werden, damit sie in der untersten Ebene liegen (sonst würden die Geister hinter diesen Objekte verschwinden). Generell gilt: Jedes neu eingefügt Objekt liegt über den vorher eingefügten Objekten.



```
public class PacWorld  extends World
{
    ….
```

```
    public void fillWorld()
    {
        int zaehler;

        // Food-Objekte mit Schleifen einfügen. Diese Objekte müssen übrigens als erstes
        // eingefügt werden, damit sie nicht über den anderen Objekten liegen.
        // 1. Reihe von links nach rechts
        for(zaehler = 60; zaehler <=740; zaehler = zaehler +40)
        {
            this.addObject(new Food(), zaehler, 60);
        }

        for(zaehler = 60; zaehler <=740; zaehler = zaehler +40)
        {
            this.addObject(new Food(), zaehler, 140);
        }

        for(zaehler = 60; zaehler <=740; zaehler = zaehler +40)
        {
            this.addObject(new Food(), zaehler, 220);
        }

        for(zaehler = 60; zaehler <=740; zaehler = zaehler +40)
        {
            this.addObject(new Food(), zaehler, 380);
        }

        for(zaehler = 60; zaehler <=740; zaehler = zaehler +40)
        {
            this.addObject(new Food(), zaehler, 460);
        }

        for(zaehler = 60; zaehler <=740; zaehler = zaehler +40)
        {
            this.addObject(new Food(), zaehler, 540);
        }
            …..
    }
}
```

Food-Objekte aufsammeln
(→ PacBoy_07)

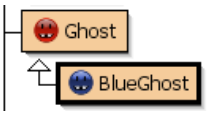
Der PacBoy soll die Food-Objekte aufsammeln (also „essen“ können). Wenn alle Objekte aufgesammelt sind, gewinnt der Spieler das Spiel. In einer Erweiterung wäre es dann schön, wenn das Spiel nicht enden würde, sondern ein neuer Level beginnt (was wir jetzt aber noch nicht umsetzen).

```
public class PacBoy  extends Actor
{
    // Wir merken uns, wie viele Früchte das Objekt "gefressen" hat:
    private int fruits = 0;

    public void act()
    {
        // Tasten prüfen:
        this.checkKeys();
        // Food-Objekte aufsammeln:
        this.eat();
        // Prüfen, ob wir gewonnen haben (alle Food-Objekte aufgesammelt):
        this.checkVictory();
        this.checkCollision();
    }

    public void eat()
    {
        Actor meinFood;
        meinFood = this.getOneIntersectingObject(Food.class);

        if(meinFood != null)
```

```
        {
            this.getWorld().removeObject(meinFood);
            Greenfoot.playSound("eat.wav");
            fruits = fruits + 1;
        }

        this.getWorld().showText("Fruits: " + fruits, 60, 20);
    }

    private void checkVictory()
    {
      // Insgesamt gibt es 108 Food-Objekte
      if(fruits == 108)
      {
      this.getWorld().addObject(new YouWin(), 400, 300);
      Greenfoot.stop();
      }
    }
}
```

Blaue Geister ändern die Richtung
(→ PacBoy_07)

Bisher weiß man immer genau, in welche Richtung sich die Ghost-Objekte bewegen werden. Wir werden das so ändern: Sobald ein Ghost-Objekt an eine Wand stößt, soll die neue Richtung „ausgewürfelt" werden. Das ist aber nur bei den beiden Geistern ganz oben und ganz unten sinnvoll (Die Geister in der Mitte würden nie mehr in die Mitte kommen, wenn Sie mal „draußen" sind, weil dafür eine Kollision nötig wäre, die aber nie stattfinden wird). Diese Geister sind also besondere Geister, verhalten sich also ein bisschen anders. Wir erstellen dafür eine neue Klasse BlueGhost, die alles von der Klasse Ghost erbt.



Im ersten Schritt ersetzen wir den ersten und den letzten Ghost durch einen BlueGhost:

```
public class PacWorld  extends World
{
    public PacWorld()
    {
        // Create a new world with 800x600 cells with a cell size of 1x1 pixels.
        super(800, 600, 1);
        fillWorld();
    }

    public void fillWorld()
    {
        …

        // ########## Geister einfügen ##########
        BlueGhost geist1;
        geist1 = new BlueGhost();
        this.addObject(geist1, 60, 60);
             …
        BlueGhost geist6;
        geist6 = new BlueGhost();
        this.addObject(geist6, 740, 540);
    }
}
```

```
public class BlueGhost extends Ghost
{
    // Dieses Attribut wird von der Super-Cass nicht vererbt, weil es private ist.
    private int xRichtung = 2;
    // Das Objekt soll sich auch nach oben und unten bewegen können:
    private int yRichtung = 0;

    // Wir haben die act-Methode von der Super-Class 'Ghost' geerbt.
    // Indem wir die Methode hier nochmal reinschreiben und verändern,
    // ÜBERSCHREIBEN wir die geerbte Methode. Überschreiben heißt:
    // Es gilt das, was hier steht, nicht die geerbte Methode!
```

```
    public void act()
    {
        this.setLocation(this.getX() + xRichtung, this.getY() + yRichtung);

        if(this.checkForWall() == true)
        {
            // Objekt einen Schritt zurücksetzen. Das heißt: Letzten schritt in x- UND y-Richtung zurücknehmen.
            this.setLocation(this.getX() - xRichtung, this.getY() - yRichtung);
            this.changeDirection();
        }
    }

    // Auswürfeln, in welche Richtung sich das Objekt bewegt:
    public void changeDirection()
    {
        // Variable Zufallszahl erstellen. Eine Variable ist eigentlich das gleiche wie ein Attribut.
        // Unterschied: Nur die Methode kennt die Variable.
        int zufallszahl;
        // Eine von 4 möglichen Zufallszahlen auswürfeln {0, 1, 2, 3}
        zufallszahl = Greenfoot.getRandomNumber(4);

        // Nach rechts
        if(zufallszahl == 0)
        {
            xRichtung = 2;
            yRichtung = 0;
        }

        // Nach links
        if(zufallszahl == 1)
        {
            xRichtung = -2;
            yRichtung = 0;
        }

        // Nach unten
        if(zufallszahl == 2)
        {
            xRichtung = 0;
            yRichtung = 2;
        }

        // Nach oben
        if(zufallszahl == 3)
        {
            xRichtung = 0;
            yRichtung = - 2;
        }
    }
}
```