Lasse Dörjer (1583997)  
Jannes Lensch (1556567)  
Yunus Ahmad (1498950)

Aufgabe 2.1

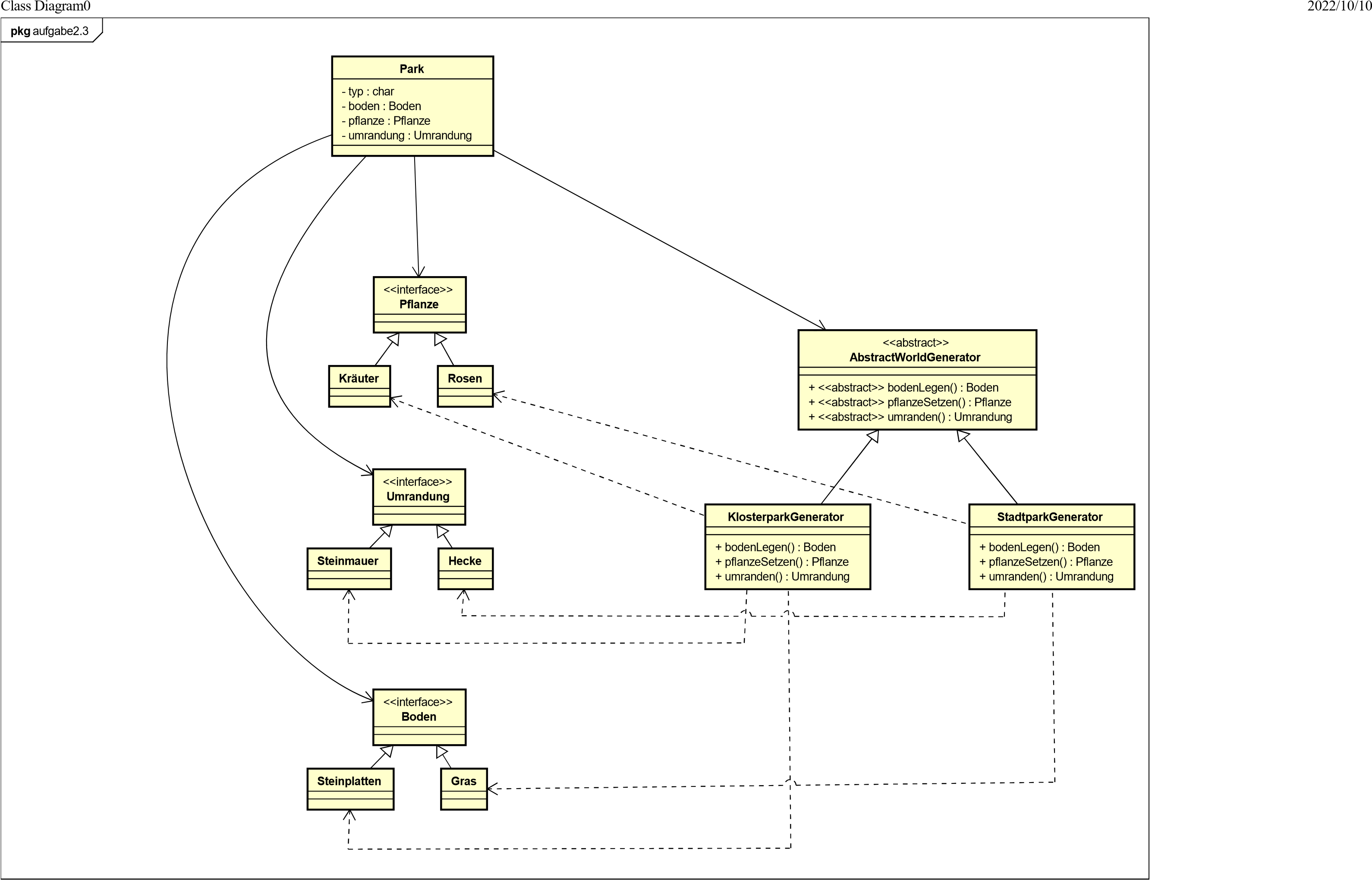
|  |  |
| --- | --- |
| Abstract | Interface |
| Instanzierbar mit Variablen | Nicht Instanzierbar 🡪 Kein Constructor |
| Nur eine Abstrakte Klasse extendbar | Mehrere implementierbar |
| Jede Art von Instanz oder statischer Variablen | Nur final static Variablen |
| Jede Art von Methoden möglich | Nur Abstrakte Methoden Default/Static Methoden (Java 8)  Private Methoden mit Implementierung (Java 9) |

* Interfaces als Vorgabe für den Aufbau von Klassen
* Abstrakte Klassen für Darstellung von Vererbung bzw. Code Wiederverwendung

Aufgabe 2.2

1. Abstrakte Oberklassen, welche in Beziehung stehen und gegenseitig Instanzen der anderen Oberklasse als Attribut besitzen. Außerdem durch Vererbung bzw. erben von diesen Oberklassen. Einfach neue Objekte hinzuzufügen, ohne viel neuen Code schreiben zu müssen.





Neuer Generator mit Methoden implementieren + neue Typen von Boden/Umrandung/Pflanzen hinzufügen

public class Park {  
 private String type;  
 private Boden boden;  
 private Pflanze pflanze;  
 private Umrandung umrandung;  
  
 public Park(String type) throws Exception {  
 this.type = type;  
  
 AbstractWorldGenerator world;  
 switch (type) {  
 case "Klosterpark":  
 world = new KlosterparkGenerator();  
 break;  
 case "Stadtpark":  
 world = new StadtparkGenerator();  
 break;  
 default:  
 throw new Exception("Typ nicht gefunden!");  
 }  
  
 boden = world.bodenLegen();  
 pflanze = world.pflanzeSetzen();  
 umrandung = world.umranden();  
 }  
}

public abstract class AbstractWorldGenerator {  
 public abstract Boden bodenLegen();  
 public abstract Pflanze pflanzeSetzen();  
 public abstract Umrandung umranden();  
}

public class KlosterparkGenerator extends AbstractWorldGenerator{  
 @Override  
 public Boden bodenLegen() {  
 return new Steinplatte();  
 }  
  
 @Override  
 public Pflanze pflanzeSetzen() {  
 return new Kräuter();  
 }  
  
 @Override  
 public Umrandung umranden() {  
 return new Steinmauer();  
 }  
}

public class StadtparkGenerator extends AbstractWorldGenerator{  
 @Override  
 public Boden bodenLegen() {  
 return new Gras();  
 }  
  
 @Override  
 public Pflanze pflanzeSetzen() {  
 return new Rosen();  
 }  
  
 @Override  
 public Umrandung umranden() {  
 return new Hecke();  
 }  
}

public interface Pflanze {  
}

public class Rosen implements Pflanze{  
}

public class Kräuter implements Pflanze {  
}

public interface Boden {  
}

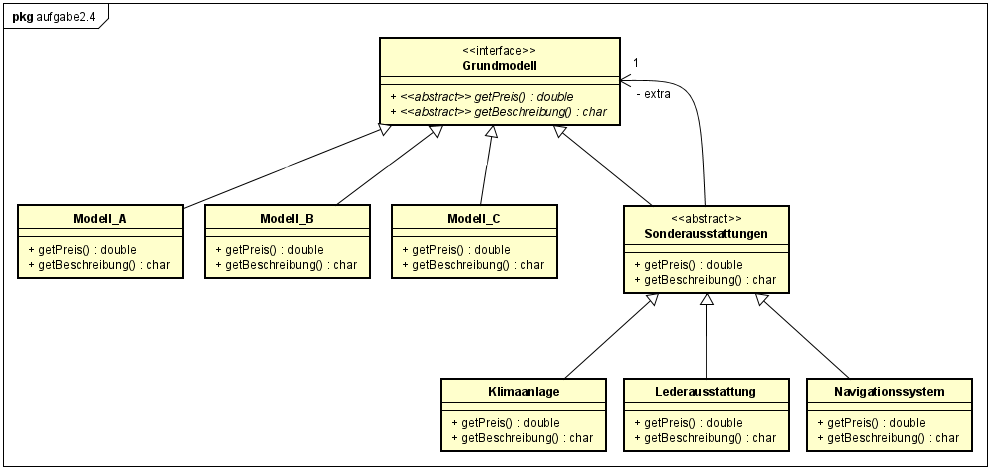
public class Steinplatte implements Boden {  
}

public class Gras implements Boden {  
}

public interface Umrandung {  
}

public class Steinmauer implements Umrandung{  
}

public class Hecke implements Umrandung {  
}

Aufgabe 2.4)

public class prog {  
 public static void main(String[] args) {  
 Grundmodell auto1 = new Navigationssystem(new Klimaanlage(new Lederausstattung(new Modell\_A())));  
  
 System.out.println(auto1.getPreis());  
 System.out.println(auto1.getBeschreibung());  
 }  
}

public interface Grundmodell {  
 double getPreis();  
 String getBeschreibung();  
}

public class Modell\_A implements Grundmodell {  
 @Override  
 public double getPreis() {  
 return 10;  
 }  
  
 @Override  
 public String getBeschreibung() {  
 return "Ein Fahrzeug des Modell\_A";  
 }  
}

public class Modell\_B implements Grundmodell {  
 @Override  
 public double getPreis() {  
 return 20;  
 }  
  
 @Override  
 public String getBeschreibung() {  
 return "Ein Fahrzeug des Modell\_B";  
 }  
}

public class Modell\_C implements Grundmodell {  
 @Override  
 public double getPreis() {  
 return 30;  
 }  
  
 @Override  
 public String getBeschreibung() {  
 return "Ein Fahrzeug des Modell\_C";  
 }  
}

public abstract class Sonderausstattungen implements Grundmodell {  
 protected Grundmodell extra;  
  
 public Sonderausstattungen(Grundmodell extra) {  
 this.extra = extra;  
 }  
}

public class Klimaanlage extends Sonderausstattungen{  
  
 public Klimaanlage(Grundmodell extra) {  
 super(extra);  
 }  
  
 @Override  
 public double getPreis() {  
 return this.extra.getPreis() + 5;  
 }  
  
 @Override  
 public String getBeschreibung() {  
 return this.extra.getBeschreibung() + " und eine Klimaanlage";  
 }  
}

public class Lederausstattung extends Sonderausstattungen{  
  
 public Lederausstattung(Grundmodell extra) {  
 super(extra);  
 }  
  
 @Override  
 public double getPreis() {  
 return this.extra.getPreis() + 7;  
 }  
  
 @Override  
 public String getBeschreibung() {  
 return this.extra.getBeschreibung() + " und eine Lederausstattung";  
 }  
}

public class Navigationssystem extends Sonderausstattungen {  
  
 public Navigationssystem(Grundmodell extra) {  
 super(extra);  
 }  
  
 @Override  
 public double getPreis() {  
 return this.extra.getPreis() + 9;  
 }  
  
 @Override  
 public String getBeschreibung() {  
 return this.extra.getBeschreibung() + " und ein Navigationssystem";  
 }  
}