Kontrollfragen A

1. Was spricht fürs objektorientierte Programmieren?
   In der realen Welt gibt es Objekte, in der objektorientierte Programmierung werden ebenfalls Objekte instanziiert. Ziel ist möglichst nahe an der realen Welt zu sein.

2. Nennen Sie einen Unterschied zwischen Objekt und Klassen.
   Klasse: „Bauplan", Sie beschreibt das Objekt.
   Objekt: „Ding", Ist das eigentliche Objekt welche anhand der Klasse erstellt wird.

3. Was programmiert man als Software-Entwickler/in?
   Klassen

4. Was steht im Zentrum beim Ausführen eines Programms?
   Die Objekte werden kreiert, anschliessen mit den Objekten interagieren.

5. Finden Sie noch andere Analogien als „Kuchen backen" zu Objekt vs. Klasse?
   Lego Bauplan => Lego-Model bauen
   Ballonverpackung => einzelner Ballon auflasen, damit spielen (interagieren)

Kontrollfragen B

1. Wie lautet der Rahmen für eine Klasse „Car"?
```
public class Car
{
...
}
```

2. Wie ist der innere Teil einer Klasse aufgebaut (1,2,3)?
   1. *Instanzvariablen*
   2. *Konstruktoren*
   3. *Methoden*

3. Womit hält man Eigenschaften eines Objektes fest?
   Instanzvariablen

4. Womit wird das Verhalten eines Objektes umgesetzt?
   Das Verhalten wird mit Methoden umgesetzt.

5. Womit kann ein Objekt initialisiert werden?
   Konstruktor

6. Deklarieren Sie für die Klasse „Car" eine Instanzvariable.
```
public class Car
{
  private String farbe;
}
```

7. Wodurch ist der Zustand eines Objektes definiert?
   Die Gesamtheit der Datenwerte eines Objekts definieren seinen aktuellen Zustand. (Instanzvariablen)

8. Wie lange existieren Instanzvariablen?
   Solange wie das Objekt existiert.

9. Muss man Instanzvariablen initialisieren?
Sie sollten trotz automatischer Initialisierung explizit initialisiert werden (bessere Lesbarkeit).

10. Initialisieren Sie die Instanzvariable der Klasse „Car“ in einem Konstruktor.
```
public Car()
{
  Farbe = “blue“;
}
```

Kontrollfragen C

1. Was ermöglichen Methoden allgemein?
Ermöglichen das Verhalten der Objekte zu realisieren.
Zugriff bzw. Kommunikation/Interaktion mit Objekten.

2. Was ermöglichen sie speziell in der OOP?
Abstraktion (einfacher Methodenaufruf, komplexer Sachverhalt)
Wiederverwendung (1 Mal implementieren, n Mal wieder verwenden)

3. Wozu dient der Methodenkopf?
Spezifikation der Methoder => WAS

4. Wozu dient der Methodenrumpf?
Implementation (Variablen Deklaration, Methoden, etc) => WIE

5. Implementieren Sie eine Methode „setSpeed“.
```
public void setSpeed(float newSpeed)
{
  speed = newSpeed;
}
```

6. Umkreisen Sie die Signatur Ihrer Methode „setSpeed“.
```
setSpeed(int newSpeed)
```

7. Implementieren Sie eine Methode „getSpeed“.
```
public float getSpeed()
{
  return speed;
}
```

8. Was bewirkt eine return-Anweisung?
Gibt den Rückgabewert zurück, die Methode wird unmittelbar danach beendet.

9. Was versteht man unter einem Block?
{ }, d.h. der Block nach dem Methodenkopf.
Innerhalb eines Blocks werden Deklarationen / Anweisungen durch „;“ voneinander getrennt.

Kontrollfragen D

1. Wo deklariert man lokale Variablen?
   Innerhalb von einer Methode bzw. Block.

2. Wie werden lokale Variablen initialisiert?
   Sie müssen explizit initialisiert werden!

3. Wie steht es mit der Sichtbarkeit von lokalen Variablen?
   Sie sind innerhalb von der Methode resp. Block sichtbar.

4. Wie steht es mit der Lebensdauer von lokalen Variablen?
   Solange wie die Methode / Block ausgeführt wird.

5. Inwiefern unterscheiden sich lokale Variablen und formale Parameter?
   Lokale Variablen werden innerhalb der Methode definiert.
   Formale Parameter werden in der Signatur definiert und beim Aufruf der Methode initialisiert.

6. Was versteht man unter einem aktuellen Parameter?
   Wenn eine Methode durch aktuelle Parameter, d.h. konkrete Werte (Ausdrücke), aufgerufen wird.

7. Variablen können als aktuelle Parameter bei einem Methodenaufruf angegeben werden. Stimmt es, dass in diesem Fall deren Werte kopiert werden?
   Ja