Java Programming

Module
Classes and Objects

DHBW Duale Hochschule Baden-Württemberg Mannheim

Prof. Dr. Holger D. Hofmann

What is Software?

- "Software is the non-tangible aspect of a computer that is necessary for it to perform any function. […]"
  http://what-is-what.com
- "Computer instructions or data. Anything that can be stored electronically is software."
  www.webopedia.com
- "Software is a generic term for programs that are used by computers and other products that contain logic circuitry (i.e., embedded systems)."
  http://www.linfo.org

What is Software? (2nd try)

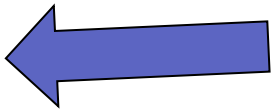
- Software allows computer users to solve a given problem
- To solve a problem, software has to model the problem
- A "problem" or situation can, e.g., be described using
  - Subjects
  - Verbs
  - Objects "Peter visits Paul"
- In Software terms,
  - Verbs are Functions
  - "Subjects" and "Objects" are Data Structures

Functions and Data Structures in C

```
struct person {
int age;
char name[25];
 } Peter, Paul, Mary;

void getOlder(struct person* p)
{ (*p).age++; // == p->age
}

Peter.name = "Peter";
Peter.age = 29;
getOlder(&Peter);
```



"Subject", "Object"???

Problem #1: Data Access

- Direct access to data structures possible
  - Using a data structure requires knowledge about its implementation
  - Changing a data structure requires changing all programs using it
- For instance, changing the age element of struct person to a date/time file would require a new implementation of void getOlder(…)

```
//Sample C Program
struct person {
int age;
char name[25];
 } Peter, Paul, Mary;

void getOlder(struct person* p)
{ (*p).age++;
}

Peter.name = "Peter";
Peter.age = 29;
getOlder(&Peter);
```

Problem #2: Functions and Data Structures

- Functions and Data Structures are separated

- In real life:
  - a Person may
    - not change his/her eye color
    - be only be able to run slowly
      - however,
    - after some training may be able to run very fastly

```
//Sample C Program
struct person {
int age;
char name[25];
 } Peter, Paul, Mary;

void getOlder(struct person* p)
{ (*p).age++;
}

Peter.name = "Peter";
Peter.age = 29;
getOlder(&Peter);
```

Problem #3: Similar "Subjects", "Objects"

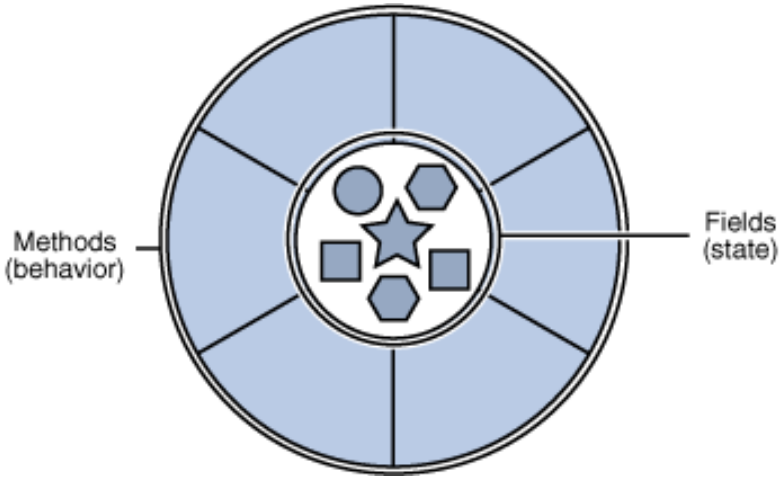
- Similar data structures cannot be treated in a similar way
  - struct student just has a StudentID element in addition to person struct
- Already existing data structures can only hardly be reused

```
//Sample C Program
struct person {
int age;
char name[25];
 } Peter, Paul, Mary;

struct student{
  int StudentID;
  int age;
  char name[25];
} Emma;
```

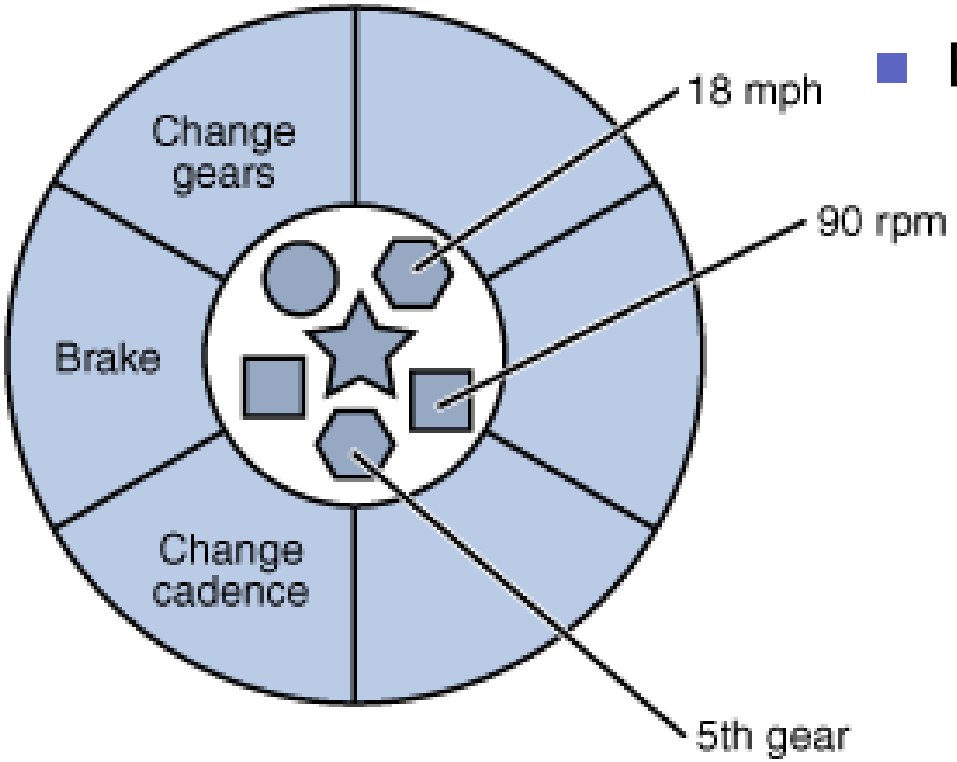
Solution: Object-Orientation

- Combines Data (Status) and Functions (Behavior)
- Basis for "modern" software development
- Provides advanced concepts for software engineering and thus for software architecture and code reuse
- Supported by programming languages such as C++, C#, Java, VB.NET, Smalltalk, Ruby



Objects

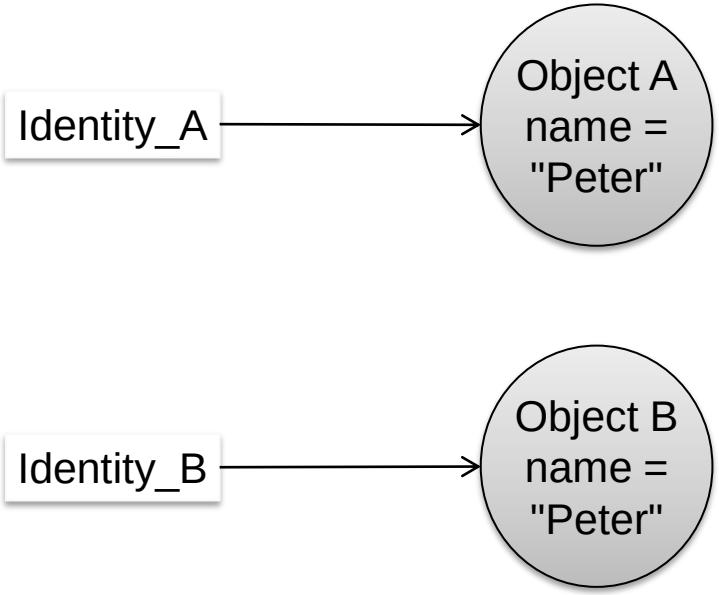
- Exist at the runtime of a program
- Encapsulate Data (Attributes) and Functions (Methods)
- Can limit access to attributes and methods, i.e., define an interface (-> Encapsulation)
- The attribute values of an object is called its "state"



- Example: Car Object

Object Identity

- each object has an identity



is Identity_A == Identity_B?

is object A == object B?

Java Example

clsPerson.java | Hello.java

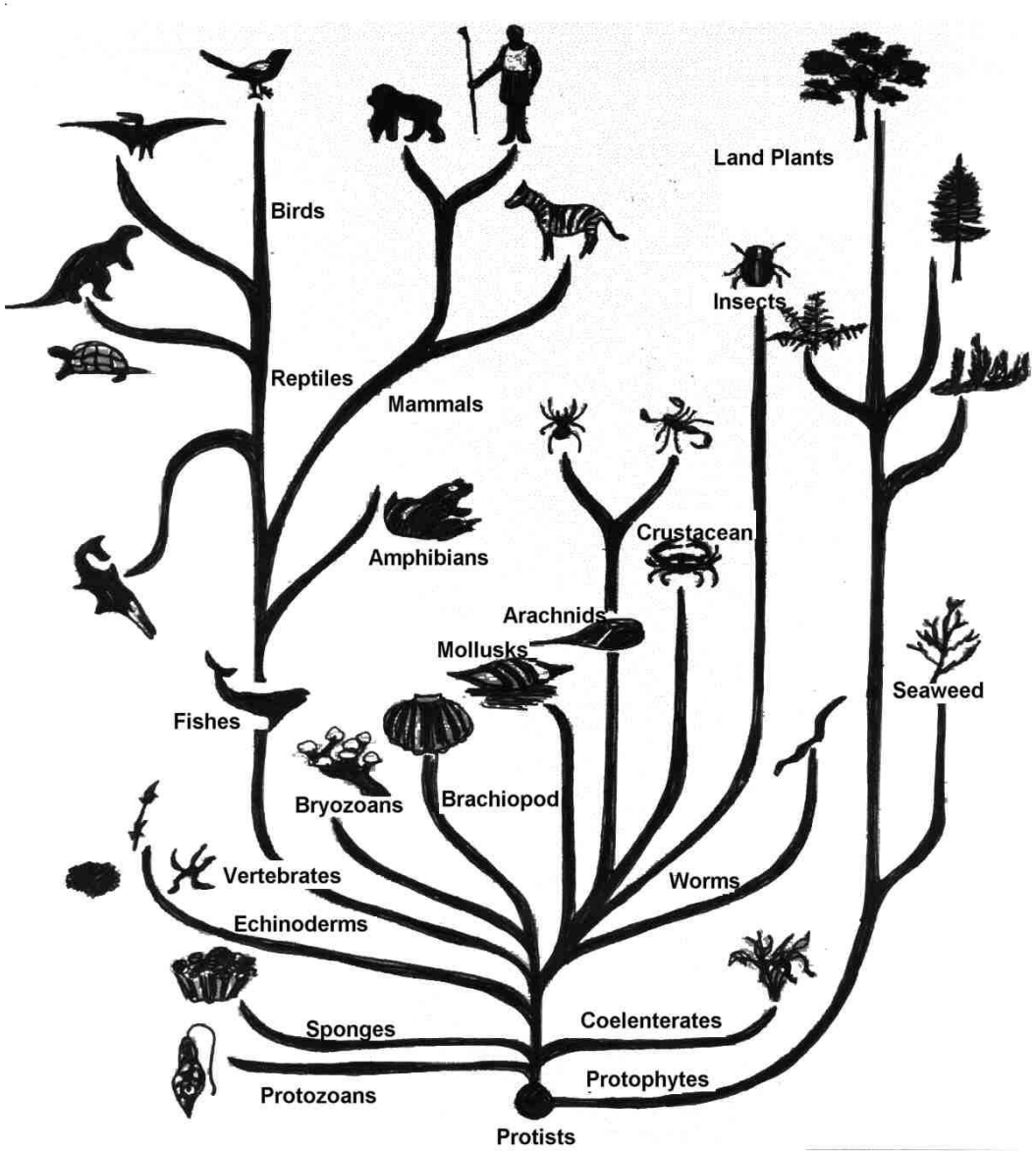
```
1  package HelloPkg;
2
3  public class clsPerson {
4      private String m_sName;
5
6      public String getName()
7      {
8          return m_sName;
9      }
10
11     public void setName(String newName)
12     {
13         m_sName = newName;
14     }
15 }
16
```

clsPerson.java | Hello.java

```
1  package HelloPkg;
2
3  public class Hello {
4      public static void main(String[] args)
5      {
6          clsPerson myPerson = new clsPerson();
7          myPerson.setName("Peter");
8
9          System.out.println(myPerson.getName());
10     }
11
12 }
13
```

Object Grouping: Classes

- Grouping of Objects to Classes according to common traits (attributes and methods)
- Example: all birds have wings, (nearly) all birds can fly
- Class acts as a template
- Template can be specialised



Reuse and Specialisation

- Classes form Groups of Objects
  - e.g., human being and dolphins below to the class of mammals
- With software, classes exist at the time of development (= source code), objects exist at runtime (=running program)
  - Classes are immutable
  - Object can change their state (and their behavior based on their state)
- Classes can be derived from other classes (inheritance) and thereby reuse their implementation

Classes in Java

- Definition by using the keyword "class"
- In Java, each class is stored to a separate file

For example, file Person.java:

```
001 public class Person {
002     String m_name; //attribute
003     int    m_age;
004     int getAge()  //method
005     {
006         return m_age;
007     }
008     void getOlder()
009     {
010         m_age++;
011     }
012 }
```

Java Packages

- Potential Problem #1: Name unambiguity
  - e.g., class "Table" as a piece of furniture and as a database class
- Potential Problem #2: Representing cohesion

  - > Solution: Java Packages
- Packages are Namespaces within Java
- They group related classes and interfaces (will be covered later)
- A package is similar to a folder

```
1. package pkgMain;
2. //all definitions below belong to pkgMain
3. public class clsMain {
4.     public static void main(String[] args) {
5.                 //...
6.         }
7. }
```

Java Objects

- Objects are created using the new operator
- Example:

```
001 Car myCar;
002 myCar = new Car();
```

- After object creation, the variable myCar contains a reference to an object
- Declaration may be combined with initialisation:

```
Car myCar = new Car();
```

- After object creation, attribute values can be set or methods can be called:

```
001 Car myCar = new Car();
002 myCar.name = "Porsche 911";
003 myCar.startEngine();
```

Putting it together:

File Person.java

```
001 package javatest1;
002
003 public class Person {
004     String m_name;
005     int    m_age;
006     int getAge()
007     {
008         return m_age;
009     }
010     void getOlder()
011     {
012         m_age++;
013     }
014 }
```

File Main.java

```
001 package javatest1;
002
003 public class Main {
004
005  public static void main(String[]      args) {
006         Person p = new Person();
007         p.m_name = "Peter";
008         p.m_age =  29;
009         System.out.println("Age before: " + p.getAge());
010         p.getOlder();
011         System.out.println("Age after: " + p.getAge());
012     }
013
014 }
```

Exercise 2.1