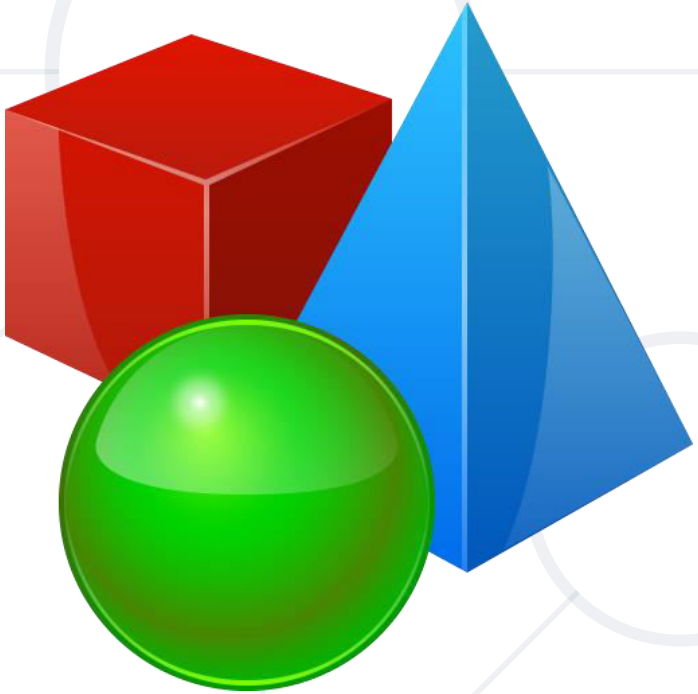
Defining Classes

Classes, Fields, Constructors, Methods



SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Contents

Have a Question?

SoftUni Foundation

sli.do

#java-advanced

Defining Classes

Defining Simple Classes

- Specification of a given type of objects from the real-world
- **Classes** provide structure for describing and creating objects

Keyword

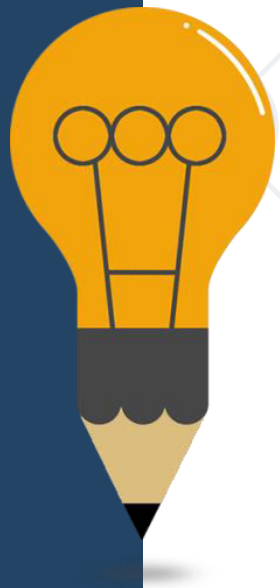
Class name

```
class Car {
    …
}
```

Class body

Naming Classes

- Use PascalCase naming
- Use descriptive nouns
- Avoid ambiguous names



```
class Dice { … }
class BankAccount { … }
class IntegerCalculator { … }
```

```
class TPMF { … }
class bankaccount { … }
class numcalc { … }
```

Class Members

- Class is made up of **state** and **behavior**
- Fields **store state**
- Methods **describe behaviour**

```
class Car {
  String make;
  String model;
  void start(){ … }
}
```

Fields

Method

```
class Dog {
  int age;
  String type;
  void bark(){ … }
}
```

Fields

Method

Creating an Object

- A class can have **many instances** (objects)

```
class Program {
  public static void main()
  {
    Car firstCar = new Car();
    Car secondCar = new Car();
  }
}
```

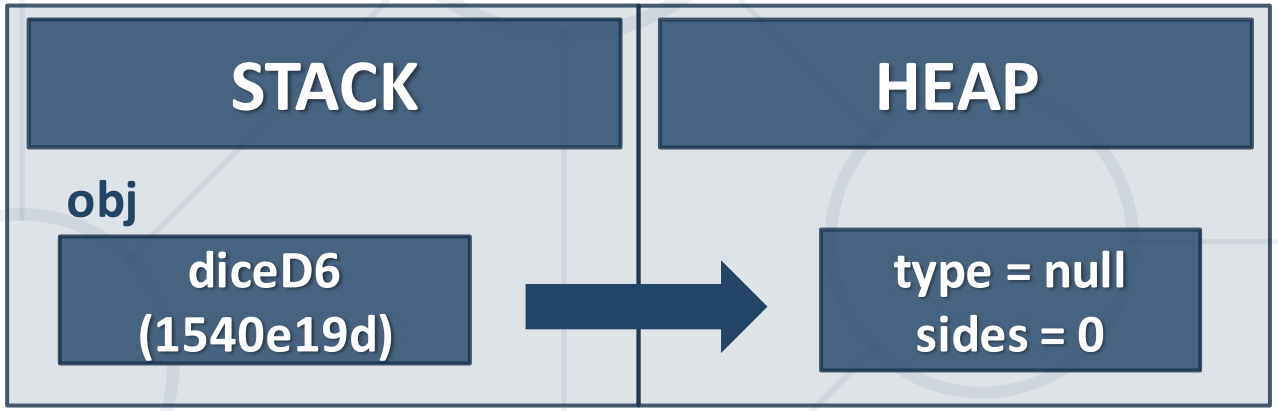
Variable stores a reference

Use the new keyword

Object Reference

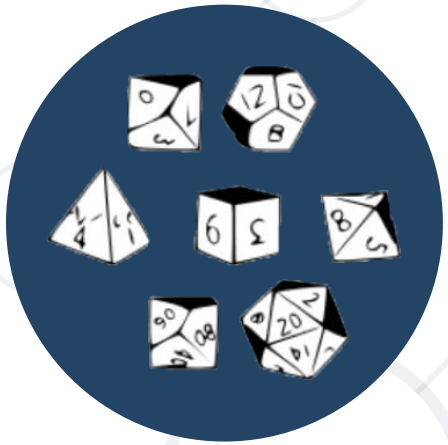
- Declaring a variable creates a **reference** in the stack
- The **new** keyword allocates memory on the heap

```
Car car = new Car();
```

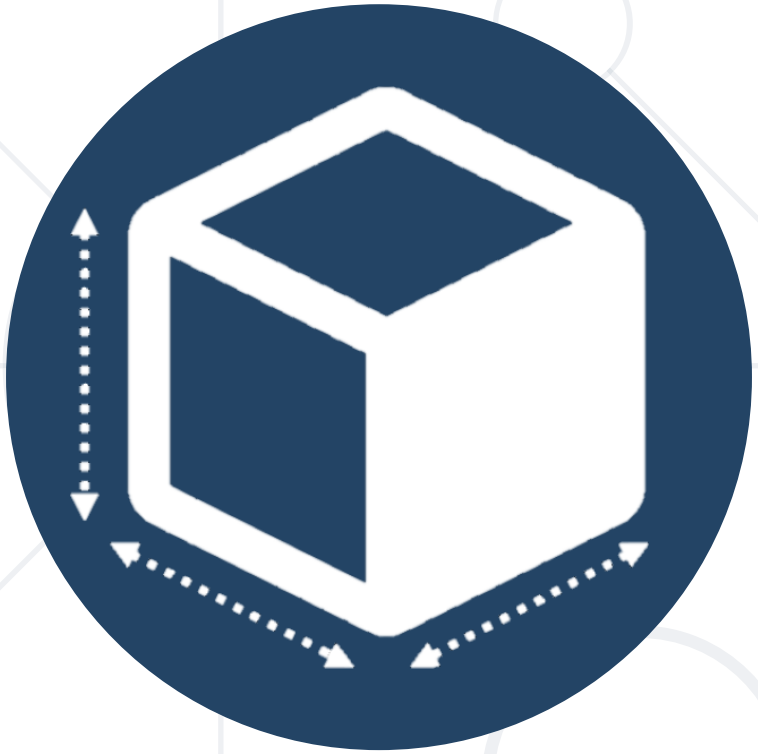


Classes vs. Objects

- Classes provide **structure** for describing and creating objects
- An **object** is a **single instance of a class**



Dice (Class) → D6 Dice (Object)



Class Data

Storing Data Inside a Class

Fields

- Class fields have **access modifiers**, **type** and **name**

```
public class Car {
  private String make;
  private int year;
  public Person owner;
  …
}
```

access modifier

type

name

Fields can be of **any type**

Problem: Define Car Class

- Create a class **Car**

```
Car
```

Class name

```
+make:String
+model:String
+horsePower:int
```

Class fields

```
(no actions)
```

Class methods

```java
public class ClassesLab {
    public static void main(String[] args) {

        Car car = new Car();

        car.make = "Chevrolet";
        car.model = "Impala";
        car.horsePower = 390;

        System.out.println(String.format(
                "The car is: %s, %s - %d HP",
                car.make, car.model, car.horsePower
        ));
    }
}
```

- Ensure proper naming!

Solution: Define Car Class

```
public class Car {
  String make;
  String model;
  int horsePower;
}
```

Access Modifiers

- Classes and class members **have modifiers**
- Modifiers **define visibility**

Class modifier

```
public class Car {
  private String make;
  private String model;
}
```

Member modifier

Fields should always be private!

Methods

Defining a Class Behaviour

Methods

- Store **executable code** (algorithm) that manipulate state

```
class Car {
  private int horsePower;

  public void increaseHP(int value) {
    horsePower += value;
  }
}
```

Getters and Setters

SoftUni Foundation

- Used to create **accessors** and **mutators** (**getters** and **setters**)

```
class Car {
  private int horsePower;
  public int getHorsePower() {
    return this.horsePower;
  }

  public void setHorsePower(int horsePower) {
    this.horsePower = horsePower;
  }
}
```

Field is hidden

Getter provides access to field

this points to the current instance

Setter provide field change

Getters and Setters

SoftUni Foundation

- Keyword **this**
  - Prevent field hiding
  - Refers to a current object

```
private int horsePower;
public void setSides(int horsePower) {
  this.horsePower = horsePower;
}

public void setSidesNotWorking(int horsePower) {
  horsePower = horsePower;
}
```

Problem: Car Info

- Create a class **Car**

- == private

Car
-make:String …
+setMake():void +getMake():String … +carInfo():String

return type

+ == public

```java
public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Car car = new Car();

        car.setMake("Chevrolet");
        car.setModel("Impala");
        car.setHorsePower(390);

        System.out.println(car.carInfo());
    }
}
```

Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab

Solution: Car Info

```
public class Car {
  private String make;
  private String model;
  private int horsePower;
  public void setMake(String make) { this.make = make; }
  public String getMake() { return this.make; }
  public String carInfo() {
    return String.format("The car is: %s %s - %d HP.",
    this.make, this.model, this.horsePower);
  }
  //TODO: Create the other Getters and Setters
}
//TODO: Test the program
```

Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab

Constructors

Object Initialization

Constructors

- Special methods, executed during object creation
- The only one way to **call a constructor** in Java is through the **keyword new**

```
public class Car {
  private String make;
  public Car() {
    this.make = "BMW";
  }
}
```

Overloading default constructor

Constructors (1)

SoftUni Foundation

- Special methods, executed during object creation

```
class Car {
  private String make;
  …
  public Car() {
    this.make = "unknown";
    …
  }
}
```

Overloading default constructor

Constructors (2)

- You can have multiple constructors in the same class

```
public class Car {
  private int horsePower; private String make;

  public Car(String make) {
    this.make = make;
  }

  public Car(String make, int horsePower) {
    this.make = make;
    this.horsePower = horsePower;
  }
}
```

Constructor with one parameter

Constructor with all parameters

Object Initial State

- Constructors **set object's initial state**

```
public class Car {
  String make;
  List<Part> parts;

  public Car(String make) {
    this.make = make;
    this.parts = new ArrayList<>();
  }
}
```

Always ensure correct state

Constructor Chaining

- Constructors can **call each other**

```
class Car {
  private String make;
  private int horsePower;

  public Car(String make, int horsePower) {
    this(make);
    this.horsePower = horsePower;
  }

  public Car(String make) {
    this.make = make;
  }
}
```

Problem: Constructors

- Create a class **Car**

Car
-make:String -model:String -horsePower:int
+Car(String make) +Car(String make, String model, int horsePower) +carInfo():String

```java
Car firstCar =
        new Car( make: "Chevrolet");

Car secondCar =
        new Car( make: "Chevrolet",  model: "Impala",  horsePower: 390);

System.out.println(firstCar.carInfo());
System.out.println(secondCar.carInfo());
```

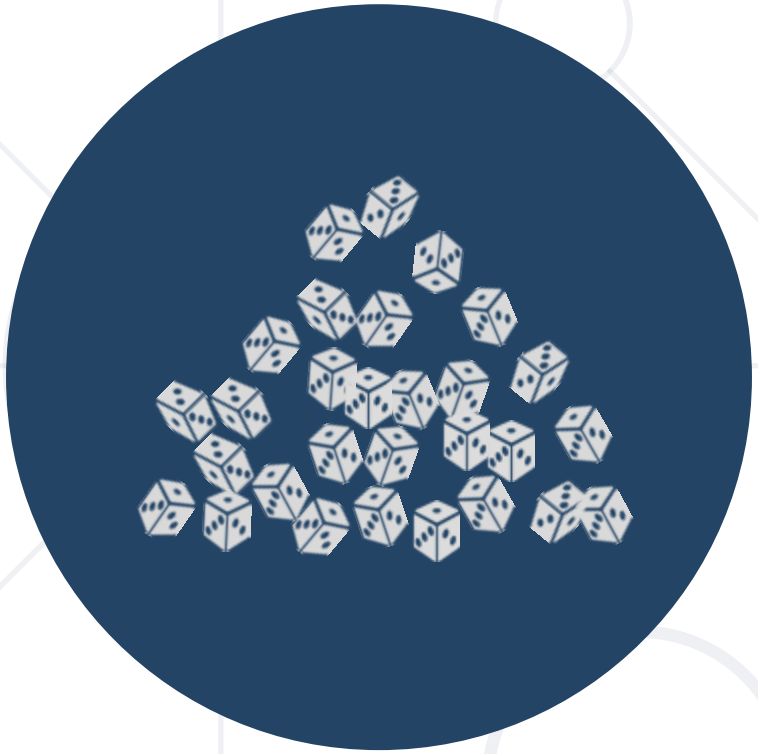
Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab

Solution: Constructors

```
public Car(String make) {
  this.make = make;
  this.model = "unknown";
  this.horsePower = -1;
}

public Car(String make, String model, int horsePower) {
  this(make);
  this.model = model;
  this.horsePower = horsePower;
}
```

Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab



Static Members

Members Common for the Class

Static Members

- Access static members **through the class name**
- Static members are **shared class-wide**
- You don't **need** an instance

```
class Program {
  public static void main(String[] args) {
    BankAccount.setInterestRate(2.2);
  }
}
```

Sets the rate for all bank accounts

Static Members

```
class BankAccount {
  private static int accountsCount;
  private static double interestRate;

  public BankAccount() {
    accountsCount++;
  }

  public static void setInterestRate(double rate) {
    interestRate = rate;
  }
}
```

Problem: Bank Account

- Create a class **BankAccount**
- Support **commands**:
  - **Create**
  - **Deposit** {ID} {Amount}
  - **SetInterest** {Interest}
  - **GetInterest** {ID} {Years}
  - **End**

BankAccount

```
-id:int (starts from 1)
-balance:double
-interestRate:double (default: 0.02)

+setInterest(double interest):void
+getId():int
+getInterest(int years):double
+deposit(double amount):void
```

underline == static

```
Create
Deposit 1 20
GetInterest 1 10
End
```

```
Account ID1 Created
Deposited 20 to ID1
4.00
```

(20 * 0.02) * 10

Solution: Bank Account

```
public class BankAccount {
  private final static double DEFAULT_INTEREST = 0.02;

  private static double rate = DEFAULT_INTEREST;
  private static int bankAccountsCount;

  private int id;
  private double balance;

  // continue…
```

Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab

Solution: Bank Account (2)

```
  public BankAccount() {
    this.id = ++bankAccountsCount;
  }

  public static void setInterest(double interest) {
    rate = interest;
  }

  // TODO: int getId()
  // TODO: double getInterest(int years)
  // TODO: void deposit(double amount)
  // TODO: override toString()
}
```

Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab

Solution: Bank Account (2)

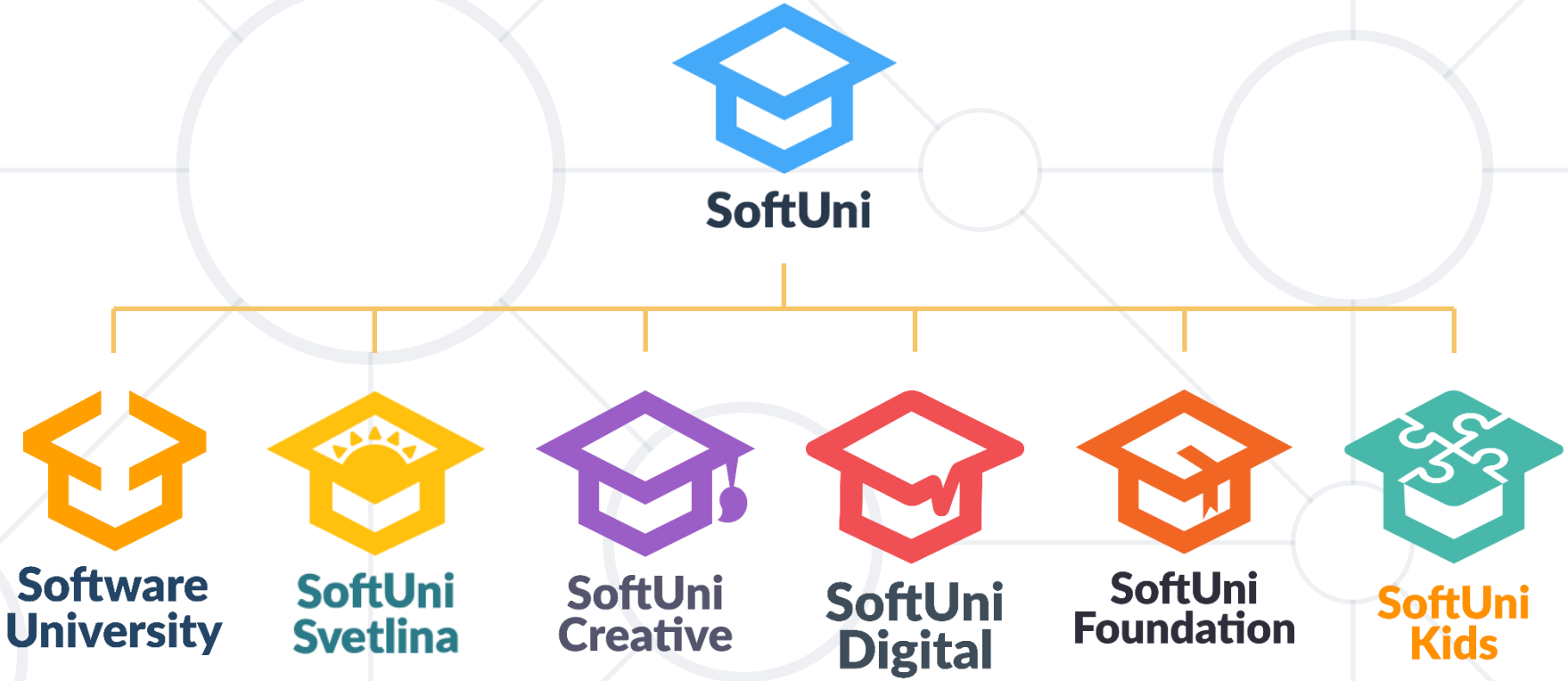
```
HashMap<Integer, BankAccount> bankAccounts = new HashMap<>();
while (!command.equals("End")) {
  //TODO: Get command args
  switch (cmdType) {
    case "Create":      // TODO
    case "Deposit":     // TODO
    case "SetInterest": // TODO
    case "GetInterest": // TODO
  }
  //TODO: Read command
}
```

Check your solution here: https://judge.softuni.bg/Contests/1517/Defining-Classes-Lab

Summary

- Classes define specific **structure** for objects
  - Objects are particular **instances of a class**
- Classes define **fields, methods, constructors** and other members
- Constructors are **invoked** when creating new class instances
- Constructors **initialize** the **object's initial state**

Questions?



https://softuni.bg/modules/59/java-advanced

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS®

STEMO®
Computer Systems & Software

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license