

Java OOP Overview

Classes and Objects, Members and Class Definition, Access Modifier, Encapsulation



SoftUni Team
Technical Trainers
Software University
<http://softuni.bg>



*Java OOP
Overview*



Table of Contents

1. Defining Simple Classes
2. Methods and Properties
3. Constructors
4. Access modifiers
5. Static Members
6. Encapsulation



Questions



sli.do

#db-advanced

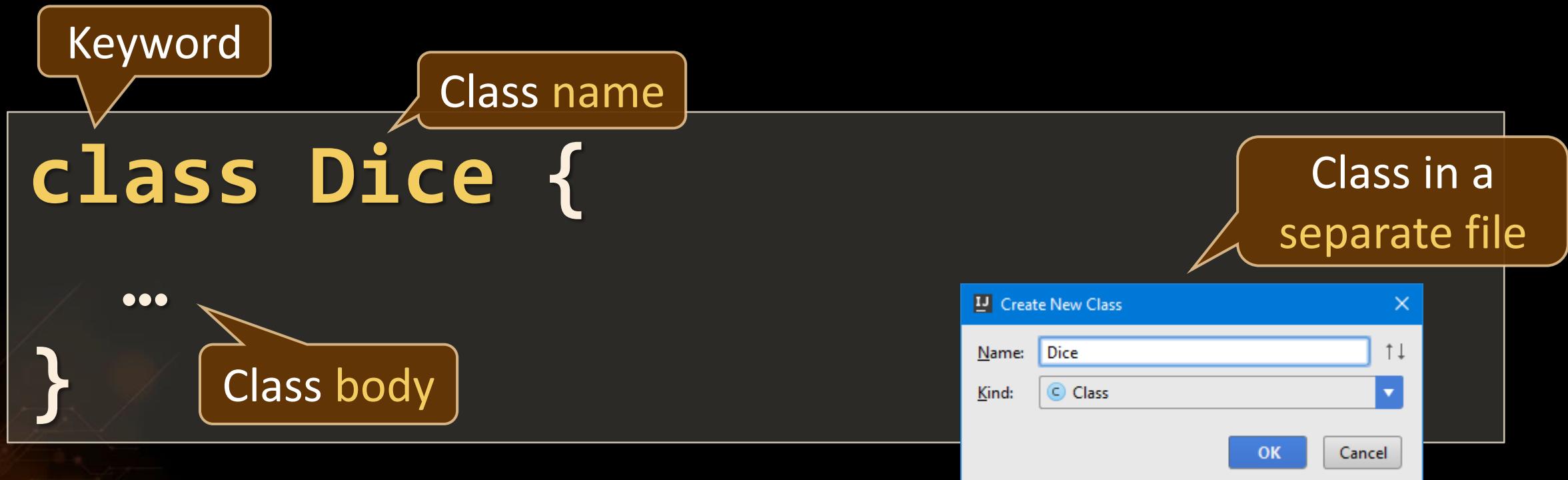


Defining Classes

Creating Class

Defining Simple Classes

- Class is a concrete implementation
- Classes provide structure for describing and creating objects



The diagram illustrates the components of a class definition and how to create a new class.

class Dice { ... **}**

- Keyword:** `class`
- Class name:** `Dice`
- Class body:** The code block between the braces, containing an ellipsis (`...`)

Create New Class

Name:	<input type="text" value="Dice"/>
Kind:	<input checked="" type="radio"/> Class

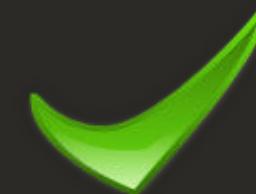
OK Cancel

A callout bubble points from the text "Class in a separate file" to the "Create New Class" dialog box.

Naming Classes

- Classes should be PascalCase
- Use descriptive nouns
- Avoid abbreviations (except widely known, e.g. URL, HTTP, etc.)

```
class Dice { ... }  
class BankAccount { ... }  
class IntegerCalculator { ... }
```



```
class TPMF { ... }  
class bankaccount { ... }  
class intcalc { ... }
```



Class Members

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

```
class Dice {  
    int sides;  
    String type;  
  
    void roll(){ ... }  
}
```

Fields

Method

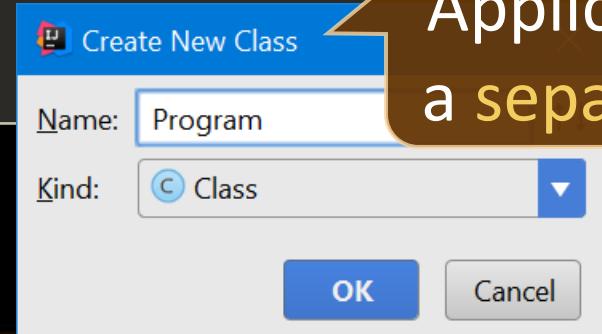
Creating an Object

- A class can have many instances (objects)

```
class Program {  
    public static void main(String args) {  
        Dice diceD6 = new Dice();  
        Dice diceD8 = new Dice();  
    }  
}
```

Variable stores
a **reference**

Use the **new**
keyword



Application in
a separate file

Object Reference

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

```
Dice diceD6 = new Dice();
```

Reference has a
fixed size

diceD6
(1540e19d)

Stack

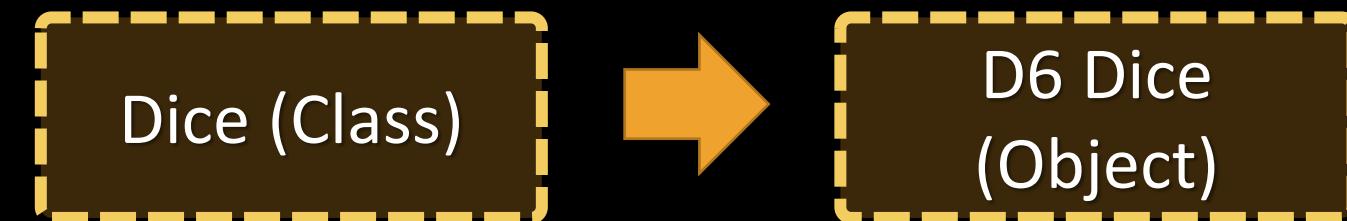
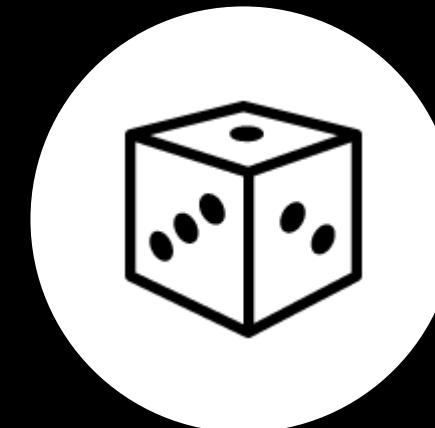
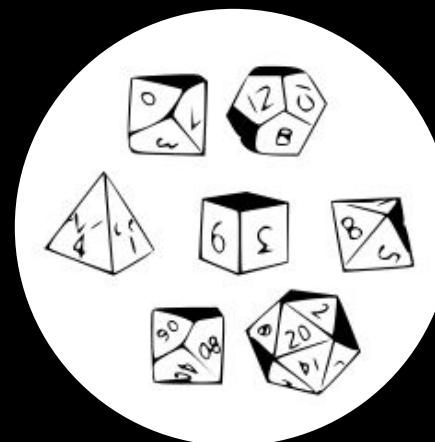
type = null
sides = 0

Heap

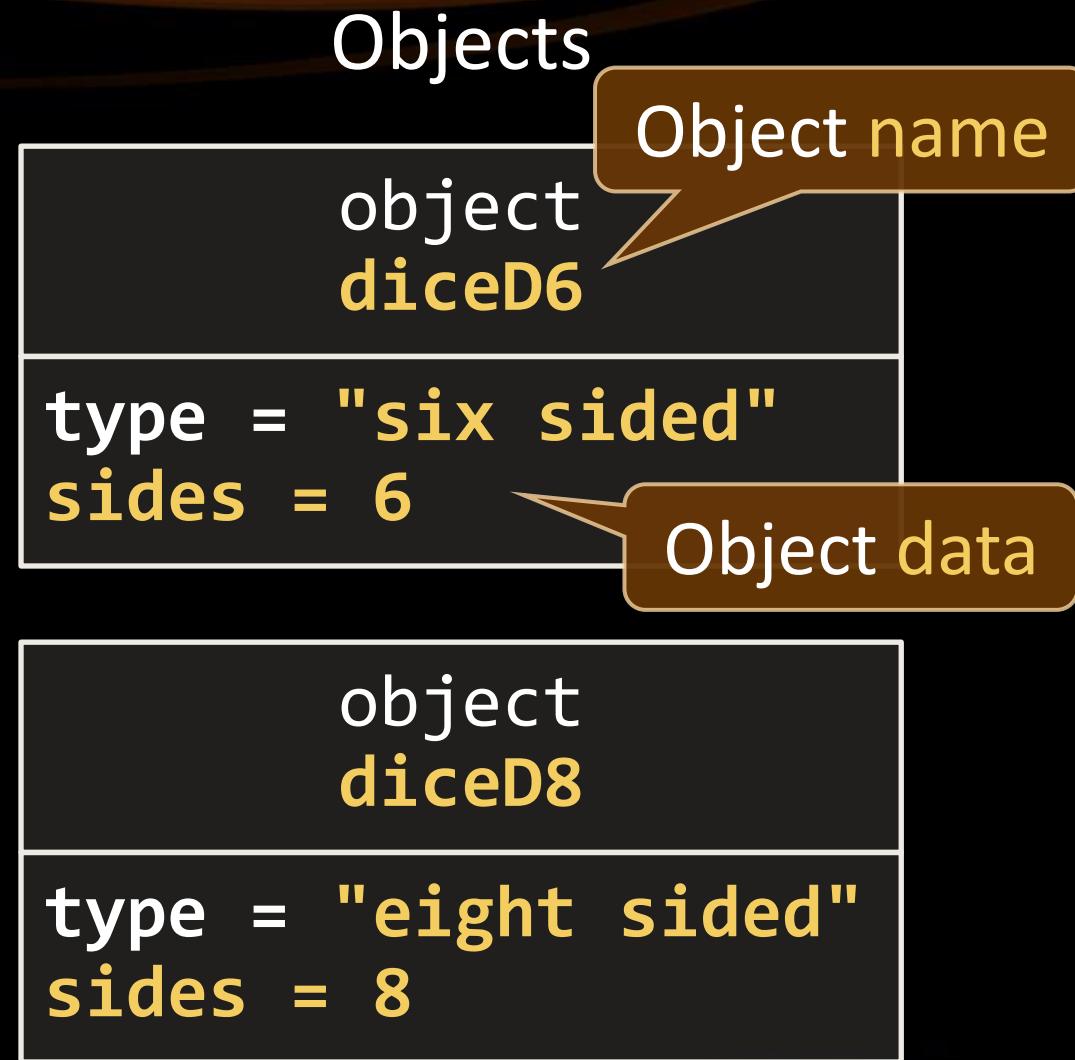
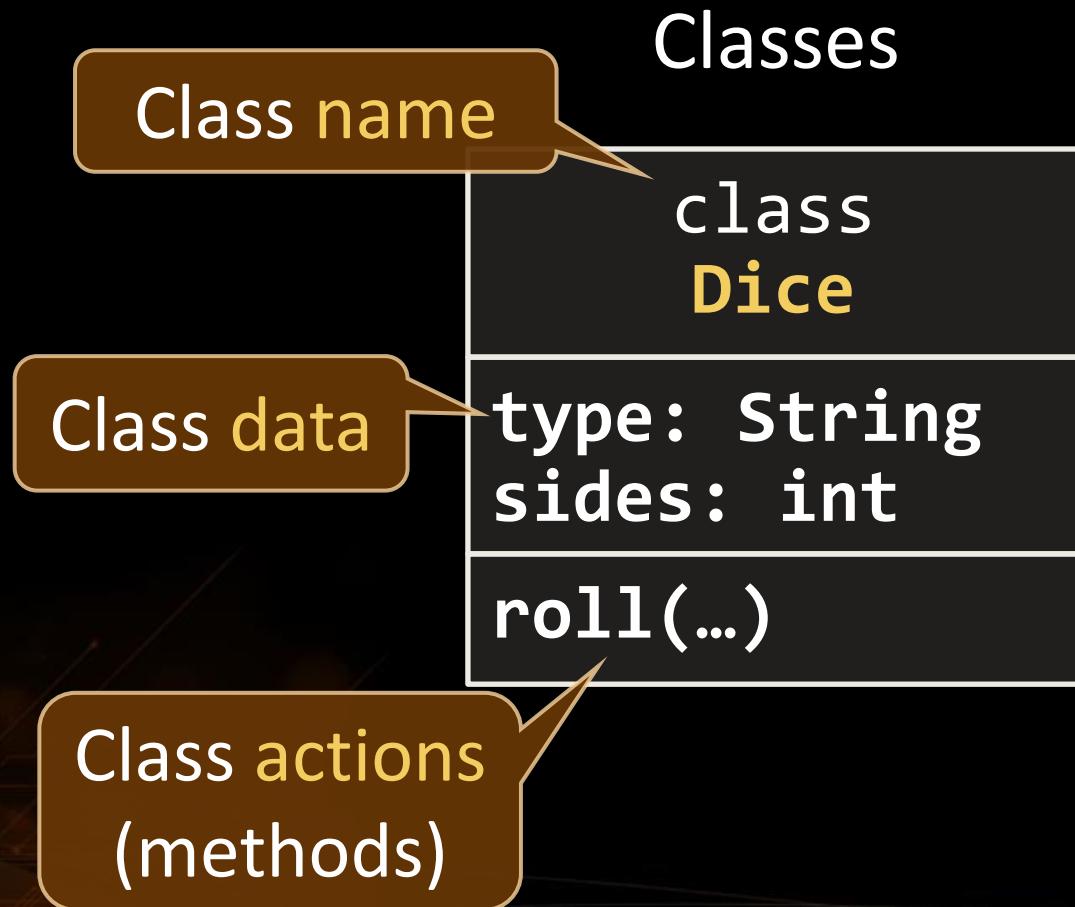
State is kept in
the heap

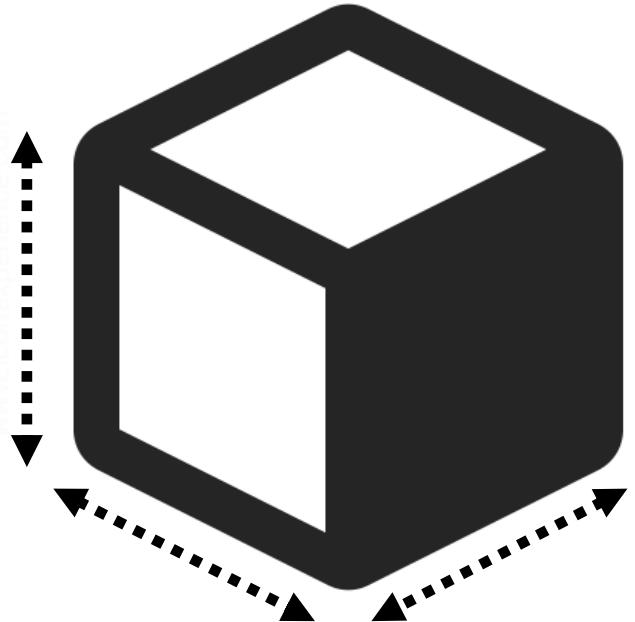
Classes vs. Objects

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



Classes vs. Objects (2)





Class Data

Storing Data Inside a Class

Fields

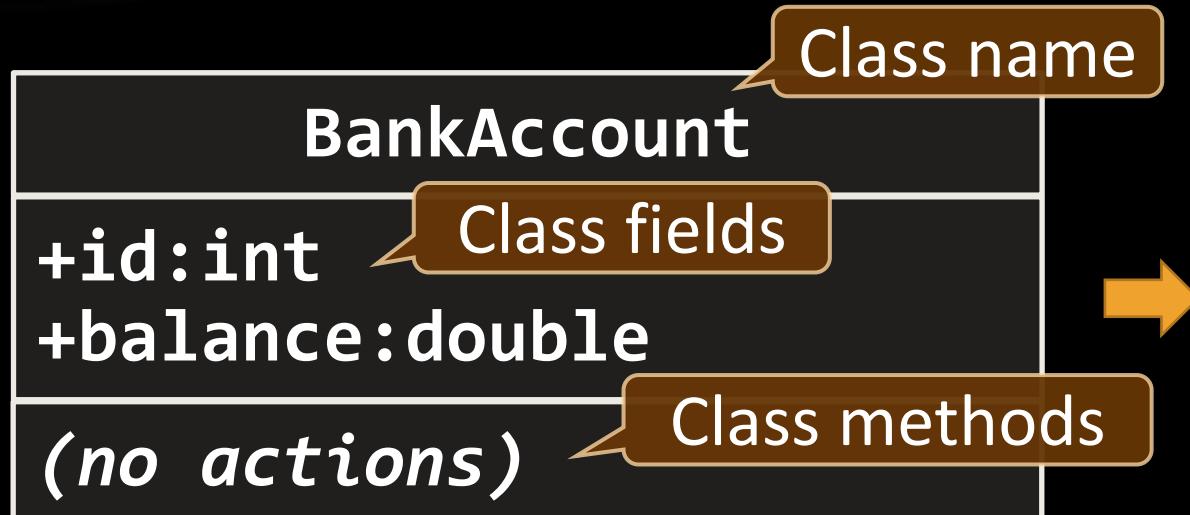
- Class fields have type and name

```
class Dice {  
    String type;  
    int sides;  
    int[] rollFrequency;  
    Person owner;  
    ...  
}
```

Fields can be
of any type

Problem: Define Class Bank Account

- Create a class **BankAccount**



```

public class Main {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount();

        acc.id = 1;
        acc.balance = 15;

        System.out.printf(
            "Account ID%d, balance %.2f",
            acc.id,
            acc.balance
        );
    }
}
  
```

The code defines a **Main** class with a **main** method. It creates a **BankAccount** object named **acc**, sets its **id** to 1 and **balance** to 15, and then prints the account details using **System.out.printf**.

- Ensure proper naming!

Solution: Define Class Bank Account

```
public class BankAccount {  
    int id;  
    double balance;  
}
```

Modifiers

- Classes and class members have modifiers
- Modifiers define visibility

Class modifier

```
public class Dice {  
    private int sides;  
    public void roll(int amount);  
}
```

Fields should
always be private!

Member modifier



Methods

Defining a Class' Behaviour

Methods

- Store executable code (algorithm) that manipulate state

```
class Dice {  
    private int sides;  
  
    public int roll() {  
        Random rnd = new Random();  
        int rollResult = rnd.nextInt(this.sides) + 1;  
        return rollResult;  
    }  
}
```

this points to the current instance

Getters and Setters

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

```
class Dice {  
    private int sides;  
    public int getSides() {  
        return this.sides;  
    }  
  
    public void setSides(int sides) {  
        this.sides = sides;  
    }  
}
```

Field is hidden

Getter provides
access to field

Setter provide
field change

Problem: Getters and Setters

- Create a class **BankAccount**

- == private

BankAccount

-id:int

-balance:double

+setId():void return type

+getBalance():double

+deposit(double amount):void

+withdraw(double amount):void

+ == public

```
public class Main {
    public static void main(String[] args) {
        BankAccount acc = new BankAccount();
        acc.setId(1);
        acc.deposit(15);
        acc.withdraw(5);

        System.out.printf(
            "Account %s, balance %.2f",
            acc,
            acc.getBalance()
        );
    }
}
```

Override
toString()

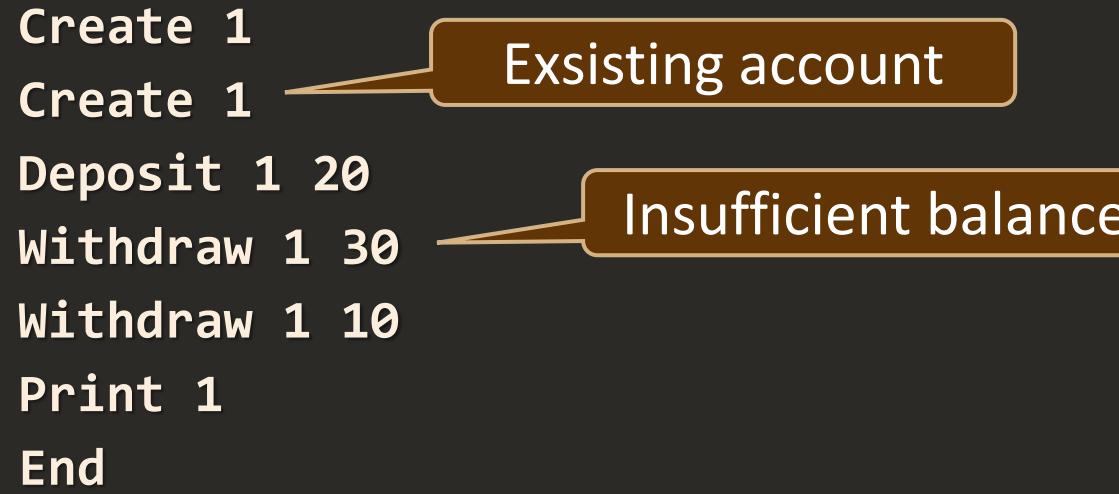
Solution: Getters and Setters

```
public class BankAccount {  
    private int id;  
    private double balance;  
  
    public void setId(int id) { return this.id; }  
    public double getBalance() { return this.balance; }  
    public void deposit(double amount) { //TODO: }  
    public void withdraw(double amount) { //TODO: }  
    @Override  
    public String toString() { return "ID" + this.id; }  
}
```

Problem: Test Client

- Create a test client that tests your **BankAccount** class
- Support commands:
 - **Create {Id}**
 - **Deposit {Id} {Amount}**
 - **Withdraw {Id} {Amount}**
 - **Print {Id}**
 - **End**

```
Create 1  
Create 1  
Deposit 1 20  
Withdraw 1 30  
Withdraw 1 10  
Print 1  
End
```



```
Account already exists .2f  
Insufficient balance  
Account ID1, balance 10.00
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/474#0>

Solution: Test Client

```
Scanner scanner = new Scanner(System.in);
HashMap<Integer, BankAccount> accounts = new HashMap<>();

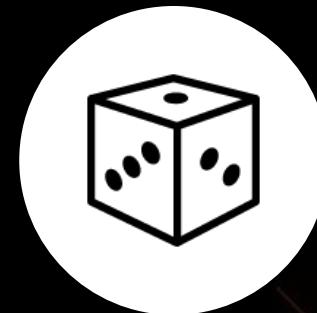
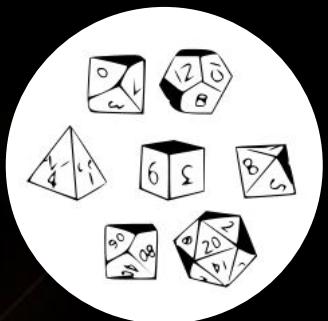
String command = scanner.nextLine();
while (!command.equals("End")) {
    // TODO: Get command arguments (cmdArgs[])
    switch (cmdType) {
        case "Create": execCreate(cmdArgs, accounts); break;
        case "Deposit": execDeposit(cmdArgs, accounts); break;
        case "Withdraw": execWithdraw(cmdArgs, accounts); break;
        case "Print": execPrint(cmdArgs, accounts); break;
    }
}
```

Solution: Test Client (2)

```
// Account creation
int id = Integer.valueOf(cmdArgs[1]);
if (accounts.containsKey(id)) {
    System.out.println("Account already exists");
} else {
    BankAccount account = new BankAccount();
    account.setId(id);
    accounts.put(id, account);
}

// TODO: Implement other commands...
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/474#0>



Defining Classes

Live Exercises in Class (Lab)



Constructors

Object Initialization

Constructors

- Special methods, executed during object creation

```
class Dice {  
    int sides;  
  
    public Dice() {  
        this.sides = 6;    }  
}
```

Overloading default constructor

Constructors (2)

- You can have multiple constructors in the same class

```
class Dice {  
    int sides;
```

```
public Dice() {  
    this.sides = 6;  
}
```

Constructor without
parameters

```
public Dice(int sides) {  
    this.sides = sides;  
}
```

Constructor with
parameters

Object Initial State

- Constructors set object's initial state

```
class Dice {  
    int sides;  
    int[] rollFrequency;  
  
    public Dice(int sides) {  
        this.sides = sides;  
        this.rollFrequency = new int[sides];  
    }  
}
```

Always ensure
correct state

Constructor Chaining

- Constructors can call each other

```
class Dice {  
    int sides;  
    public Dice() {  
        this(6);  
    }  
    public Dice(int sides) {  
        this.sides = sides;  
    }  
}
```

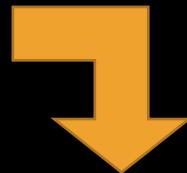
Calls constructor with parameters

6 should be declared in a final variable

Problem: Define Person Class

- Create a class Person

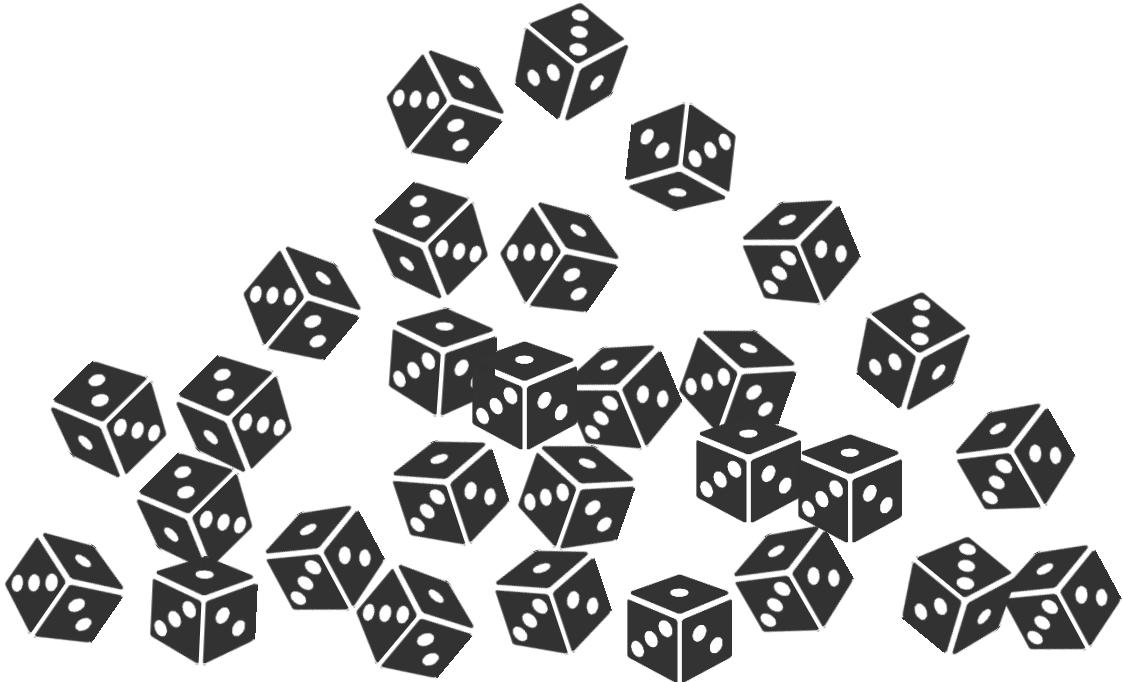
Person
<pre>-name:String -age:int -accounts:List<BankAccount></pre>
<pre>+Person(String name, int age) +Person(String name, int age, List<BankAccount> accounts) +getBalance():double</pre>



```
Person andy =  
    new Person("Andy Price", 32);  
  
Person greg =  
    new Person("Greg Mills",  
              22, new ArrayList<>());
```

Solution: Define Person Class

```
public Person(String name, int age) {  
    this.name = name;  
    this.age = age;  
    this.accounts = new ArrayList<>();  
}  
  
public Person(String name, int age, List<BankAccount> accs) {  
    this.name = name;  
    this.age = age;  
    this.accounts = accs;  
}
```



Static Members

Members Common for the Class

Static Members

- Static members are shared class-wide

```
class BankAccount {  
    private static int accountsCount;  
  
    public BankAccount() {  
        accountsCount++;  
    }  
    ...  
}
```

Static Members (2)

- Static members are shared class-wide

```
class BankAccount {  
    private static double interestRate;  
  
    public static setInterestRate(double rate) {  
        interestRate = rate;  
    }  
    ...  
}
```

Accessing Static Members

- Access static members through the class name
- 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

Problem: Static Id and Rate

- Create a class **BankAccount**

- Support commands:

- Create
- Deposit {ID} {Amount}
- SetInterest {Interest}
- GetInterest {ID} {Years}
- End

BankAccount
<u>-id:int</u> (starts from 1) <u>-balance:double</u> <u>-interestRate:double</u> (default: 0.02)
<u>+setInterest(double interest):void</u> <u>+getInterest(int years):double</u> <u>+deposit(double amount):void</u>

underline == static

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



```
Account ID1 Created
Deposited 20 to ID1
4.00
(20 * 0.02) * 10
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/474#0>

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;  
  
    // constructor and methods...  
}
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/474#0>

Solution: Bank Account (2)

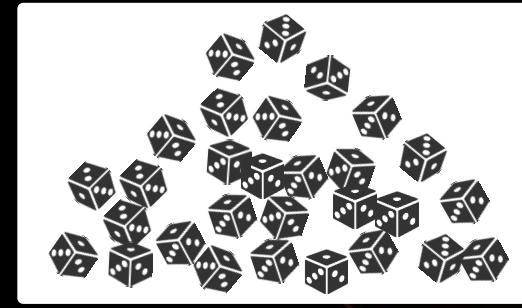
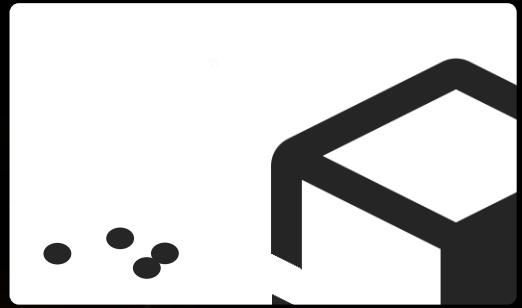
```
public class BankAccount {  
    // continued...  
  
    public BankAccount() {  
        this.id = ++bankAccountsCount;  
    }  
  
    public static void setInterest(double interest) {  
        rate = interest;  
    }  
  
    // TODO: override toString()  
    // TODO: void deposit(double amount)  
    // TODO: double getInterest(int years)  
}
```

Check your solution here: <https://judge.softuni.bg/Contests/Practice/Index/474#0>

Solution: Bank Account (2)

```
HashMap<String, 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/Practice/Index/474#0>



Constructors and Static Members

Live Exercises in Class (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



Java OOP Overview



Questions?



License

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



- Attribution: this work may contain portions from
 - "Fundamentals of Computer Programming with Java" book by Svetlin Nakov & Co. under CC-BY-SA license
 - "OOP" course by Telerik Academy under CC-BY-NC-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
 - softuni.bg
- Software University @ Facebook
 - facebook.com/SoftwareUniversity
- Software University @ YouTube
 - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg

