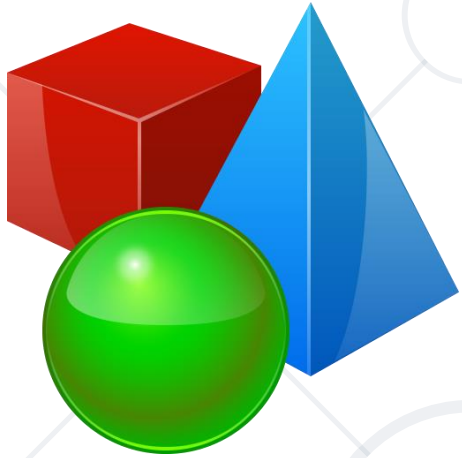


# Objects and Classes

Using Objects and Classes

Defining Simple Classes



**SoftUni Team**  
Technical Trainers



**SoftUni**



Software University

<https://softuni.bg>

[sli.do](https://sli.do)

**#fund-java**

1. **Objects**
2. **Classes**
3. **Built-in** Classes
4. **Defining** Simple Classes
  - Fields
  - Constructors
  - Methods





# Objects and Classes

# Objects

- An **object** holds a set of named values
  - E.g. **birthday** object holds the day, month, and year
  - Creating a birthday object:



**Birthday**

day = 27

month = 11

year = 1996

Object  
**name**

Object  
**fields**

```
LocalDate birthday =  
    LocalDate.of(2018, 5, 5);  
System.out.println(birthday);
```

Create a new object of  
type LocalDate

# Classes

- In programming **classes** provide the structure for **objects**
  - Act as a **blueprint** for **objects** of the same type
- Classes define:
  - **Fields (private variables)**, e.g. day, month, year
  - **Getters/Setters**, e.g. getDay, setMonth, getYear
  - Actions (**behavior**), e.g. plusDays(count), subtract(date)
- Typically, a class has multiple **instances** (objects)
  - Sample class: **LocalDate**
  - Sample objects: **birthdayPeter, birthdayMaria**



# Objects – Instances of Classes

- Creating the object of a defined class is called **instantiation**
- The **instance** is the object itself, which is created runtime
- All instances have common **behavior**



```
LocalDate date1 = LocalDate.of(2018, 5, 5);  
LocalDate date2 = LocalDate.of(2016, 3, 5);  
LocalDate date3 = LocalDate.of(2013, 3, 2);
```

# Classes vs. Objects

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







`Math.max()`

# Using the Built-In API Classes

Math, Random, BigInteger ...

- Java provides **ready-to-use** classes:
  - Organized inside Packages like:  
**java.util.Scanner**, **java.util.List**, etc.

- Using static class members:

```
LocalDateTime today = LocalDateTime.now();  
double cosine = Math.cos(Math.PI);
```

- Using non-static Java classes:

```
Random rnd = new Random();  
int randomNumber = rnd.nextInt(99);
```

# Problem: Randomize Words

- You are given a list of words
  - Randomize their order and print each word on a separate line



**Note: the output is a sample.  
It should always be different!**

Check your solution here: <https://judge.softuni.org/Contests/1319/>

# Solution: Randomize Words

```
Scanner sc = new Scanner(System.in);
String[] words = sc.nextLine().split(" ");
Random rnd = new Random();
for (int pos1 = 0; pos1 < words.length; pos1++) {
    int pos2 = rnd.nextInt(words.length);
    //TODO: Swap words[pos1] with words[pos2]
}
System.out.println(String.join(
    System.lineSeparator(), words));
```

# Problem: Big Factorial

- Calculate  $n!$  ( $n$  factorial) for very big  $n$  (e.g. 1000)

5 → 120      10 → 3628800      12 → 479001600

50 → 3041409320171337804361260816606476884437764156  
8960512000000000000

88 → 1854826422573984391147968456455462843802209689  
4939934668442158098688956218402819931910014124  
48045018284166335168512000000000000000000000000

Check your solution here: <https://judge.softuni.org/Contests/1319/>

# Solution: Big Factorial

```
import java.math.BigInteger;
...
int n = Integer.parseInt(sc.nextLine());
BigInteger f = new BigInteger(String.valueOf(1));
for (int i = 1; i <= n; i++) {
    f = f.multiply(BigInteger
        .valueOf(Integer.parseInt(String.valueOf(i))));
}
System.out.println(f);
```

Use the  
`java.math.BigInteger`

**N!**



# Defining Classes

Creating Custom 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 Dice {
```

...

```
}
```

Class **body**



# Naming Classes

- Use **PascalCase** naming
- Use **descriptive** nouns
- Avoid abbreviations (except widely known, e.g. URL, HTTP, etc.)



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

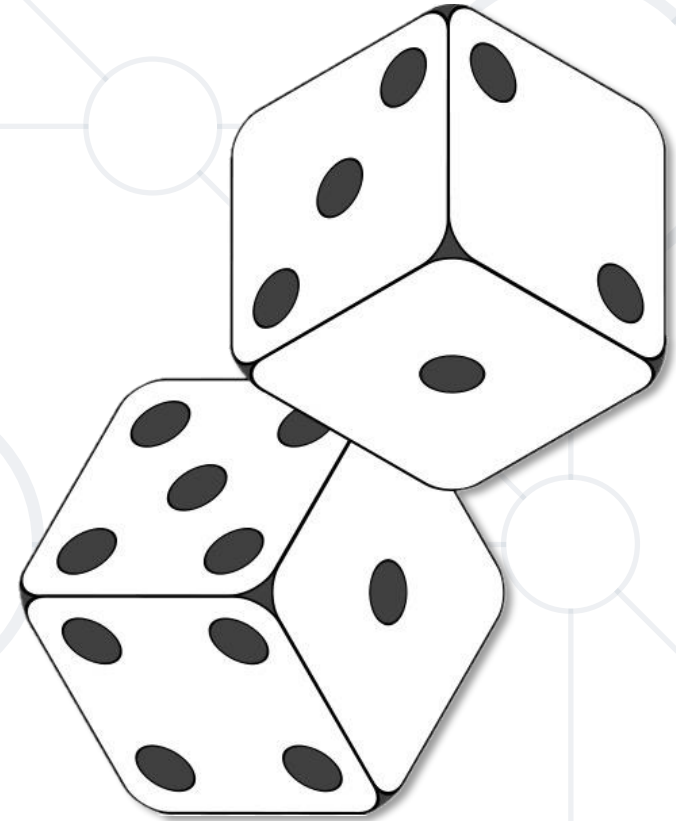
```
class TPMF { ... }  
class bankaccount { ... }  
class intcalc { ... }
```

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

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

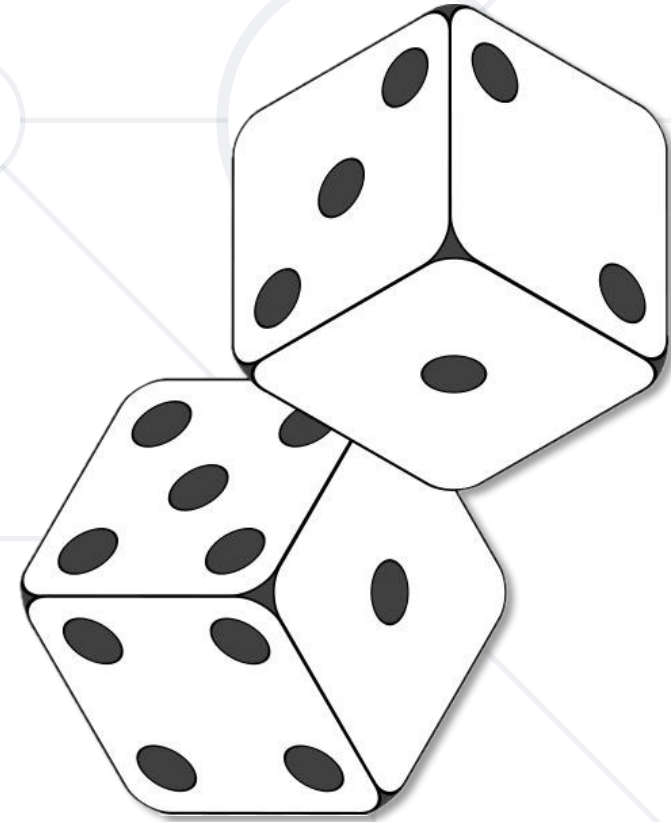
Field

Method



- Store executable code (algorithm)

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



# Getters and Setters

```
class Dice {  
    . . .  
    public int getSides() { return this.sides; }  
    public void setSides(int sides) {  
        this.sides = sides;  
    }  
    public String getType() { return this.type; }  
    public void setType(String type) {  
        this.type = type;  
    }  
}
```

Getters & Setters



# 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();  
    }  
}
```

Use the **new** keyword

Variable stores a  
**reference**

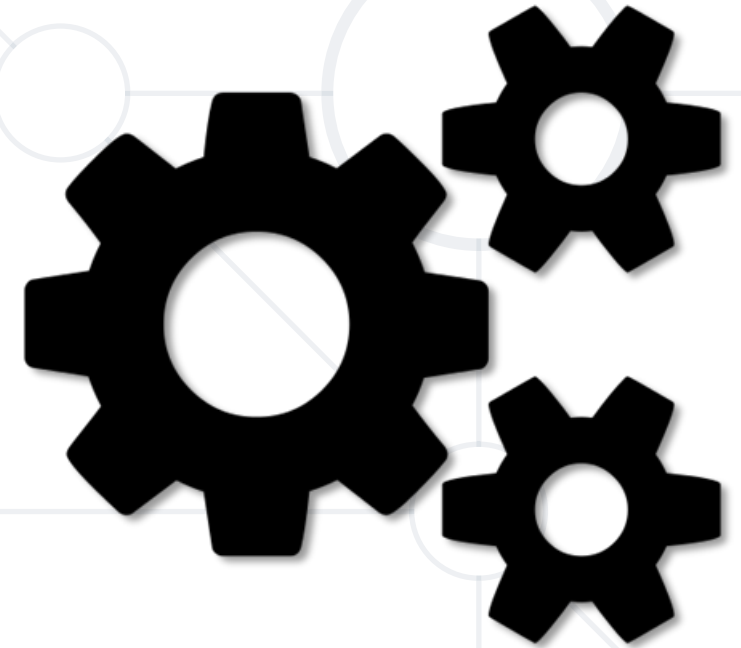


- Special methods, executed during object creation

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

**Constructor name** is  
the same as the  
name of the class

**Overloading** default  
constructor



- You can have multiple constructors in the same class

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

```
class StartUp {  
    public static void main(String[] args) {  
        Dice dice1 = new Dice();  
        Dice dice2 = new Dice(7);  
    }  
}
```

- Read students until you receive "**end**" in the following format:
  - "**{firstName} {lastName} {age} {hometown}**"
  - Define a class **Student**, which holds the needed information
  - If you receive a student which already exists (matching **firstName** and **lastName**), overwrite the information
- After the end command, you will receive a city name
- Print students which are from the given city in the format:  
"**{firstName} {lastName} is {age} years old.**"



```
public Student(String firstName, String lastName,  
                int age, String city){  
    this.firstName = firstName;  
    this.lastName = lastName;  
    this.age = age;  
    this.city = city;  
    // TODO: Implement Getters and Setters  
}
```

```
List<Student> students = new ArrayList<>();
String line;
while (!line.equals("end")) {
    // TODO: Extract firstName, LastName, age, city from the input
    Student existingStudent = getStudent(students, firstName, lastName);
    if(existingStudent != null) {
        existingStudent.setAge(age);
        existingStudent.setCity(city);
    } else {
        Student student = new Student(firstName, lastName, age, city);
        students.add(student);
    }

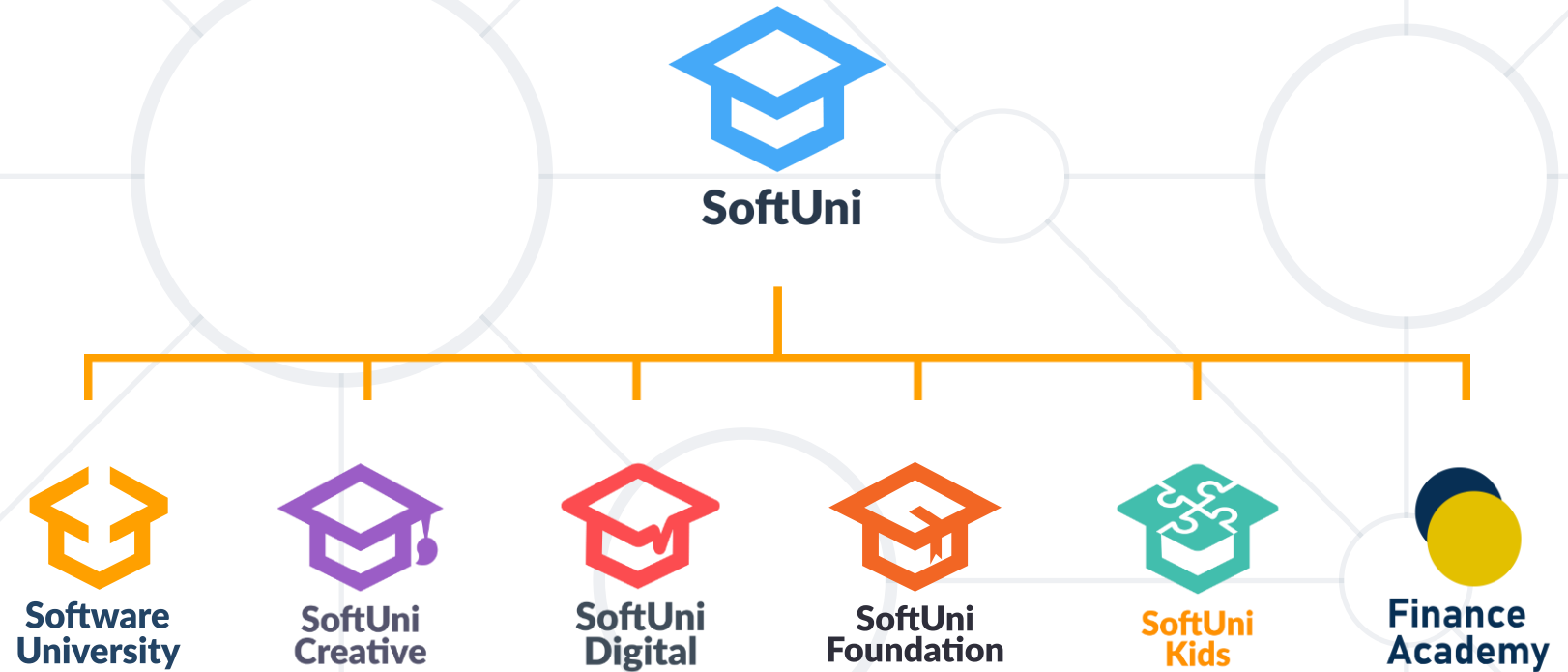
    line = sc.nextLine();
}
```

```
static Student getStudent(List<Student> students, String firstName,  
                           String lastName) {  
    for (Student student : students){  
        if(student.getFirstName().equals(firstName)  
            && student.getLastName().equals(lastName))  
            return student;  
    }  
  
    return null;  
}
```

- Classes define templates for object
  - **Fields**
  - **Constructors**
  - **Methods**
- Objects
  - Hold a set of **named values**
  - **Instance** of a class



# Questions?



# SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>

