Welcome to the

Java Course

Module 3 – Day 02

Content of the course

- Object-Oriented Programming concepts
- Classes and objects
- **Inheritance and polymorphism**
- Encapsulation and accessibility
- Exceptions

Air Flight Company Project - Step 1

- Create a “Flight” class with properties to store the following information:
  - Flight number
  - Destination
  - Capacity
  - Amount of booked seats

- All properties should be private and the class should have get and set methods for them.

Air Flight Company Project - Step 1

- Whenever a new Flight gets created, it should have no booked seats.

- The “Flight” class should provide a method to allow booking a seat. This method will return true if it was possible to book a seat and will increment the amount of booked seats for the flight. It will return false if it was not possible to book a seat because the flight is already full.

Air Flight Company Project - Step 1

```
>>> New Flight <<<
Enter flight number: 3527
Enter destination: Madrid
Enter flight capacity: 180
Flight created.

Would you like to (a) book a seat or (b) see the amount of available
seats? a
Seat booked!
Would you like to (a) book a seat or (b) see the amount of available
seats? a
Seat booked!
Would you like to (a) book a seat or (b) see the amount of available
seats? b
Available seats on flight 3527 to Madrid: 178
```

ENUM (enumeration)

```
breed = DogBreads.CHIHUAHUA;
```

- CONSTANTS

Commonly used for :

- choices
- command options
- states
- modes

```
public enum DogBreed {
  LABRADOR,
  BEAGLE,
  BULLDOG,
  DACHSHUND,
  GERMAN_SHEPHERD,
  GOLDEN_RETRIEVER,
  PUG,
  ROTTWEILER,
  SIBERIAN_HUSKY,
  CHIHUAHUA
  // Add more breeds as needed
}
```

Enums can have attributes and methods

```
public enum Season {
    WINTER("Cold"), SUMMER("Hot"), SPRING("Warm"), FALL("Cool");

    private String attribute;

    // Constructor
    Season(String attribute) {
        this.attribute = attribute;
    }

    public String getAttribute() {
        return attribute;
    }
}
```

Example of switch using Enum

```
public enum Day {
    MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY
}
```

```
Day day = Day.MONDAY;

switch (day) {
    case MONDAY:
        System.out.println("Mondays are tough!");
        break;
    case FRIDAY:
        System.out.println("Fridays are better!");
        break;
    default:
        System.out.println("Midweek days are so-so.");
        break;
}
```

Now YOUR TURN !

Let’s do exercise 1

Inheritance

Now we know what a class is. For example, when we talk about the "*Animal*" class, we aren't referring to a specific animal but to the **generic and abstract concept** with attributes and actions.

Inheritance

```
public class Dog {
  private String name;
  private int age;
  private DogBreads breed;
}
```

```
public class Cat {
  private String name;
  private int age;
  private CatBreads breed;
}
```

Inheritance

```
public class Dog {
  private String name;
  private int age;
  private DogBreads breed;
}
```

```
public class Cat {
  private String name;
  private int age;
  private CatBreads breed;
}
```

Inheritance

PARENT

```
public class CompanyAnimal {
  private String name;
  private int age;
}
```

CHILD

```
public class Dog extends
CompanyAnimal {
  private DogBreads breed;
}
```

CHILD

```
public class Cat extends
CompanyAnimal {
  private CatBreads breed;
}
```

Inheritance

- The **Parent** has no secret from the child's class.
- A class can have **ONE** parent, but a parent can have **MANY** children.

Inheritance

```
public class CompanyAnimal {
  private String name;
  private int age;

  public CompanyAnimal(){
    this.name = “”;
    this.age = “”;
  }
}
```

PARENT

```
public class Dog extends CompanyAnimal {
  private DogBreads breed;

  public Dog(){
    super();
    this.breed = DogBreads.CHIHUAHUA;
  }

}
```

CHILD

Override

- The **Parent** class had methods, those methods can be overridden in the child class.
- All classes extend **Object** parent by default, this class has methods that can be used or overridden

Override

```
public class Dog {
  private String name;
  private int age;

  public Dog(String name, int age) {
    this.name = name;
    this.age = age;
  }

  @Override
  public String toString() {
    return "Dog{" +
    "name='" + name + '\'' + ", age=" + age + '}';
  }
}
```

Now YOUR TURN !

Let’s do exercise 2

Polymorphism

When we create a function (or method), we know we have to give it an intuitive and unique name.

However, the principle of **Polymorphism** bypasses this uniqueness limit.

Polymorphism

Thanks to Polymorphism, functions with the same name can exist as long as they handle **different parameters**.

```
public Animal() {
  this.animalType = AnimalType.UNNKOWN;
  this.alive = true;
  this.age = 0;
}

public Animal(AnimalType animalType, boolean alive, int age) {
  this.animalType = animalType;
  this.alive = alive;
  this.age = age;
}
```

Polymorphism

```
public void sum( num1, num2 ) {
  // with 2 passed parameters
}
public void sum( num1, num2, num3) {
  // with 3 passed parameters
}
```

Now YOUR TURN !

Let’s do exercise 3

Air Flight Company Project - Step 2

- Create an ENUM to represent the flight status, the statuses could be ON_TIME, DELAYED, CANCELLED
- Add a property to the “Flight” class to store the flight’s status.
- Whenever a new Flight gets created, its status should be ON_TIME

Air Flight Company Project - Step 2

- Modify the main program to allow adding multiple flights.
- The main program should also allow updating the status of a flight

Air Flight Company Project - Step 2

```
>>> New Flight <<<
Enter flight number: 3527
Enter destination: Madrid
Enter flight capacity: 180
Flight created. Would you like to add another flight (y/n)? y

>>> New Flight <<<
Enter flight number: 3017
Enter destination: Paris
Enter flight capacity: 135
Flight created. Would you like to add another flight (y/n)? n

Would you like to (a) book a seat, (b) see the amount of available seats
or (c) update a flight? c
Enter the flight number: 3017
Enter the new status (o) on-time, (d) delayed or (c) cancelled: d
```

Air Flight Company Project - Step 2

```
Would you like to (a) book a seat, (b) see the amount of available seats
or (c) update a flight? a
Enter the flight number: 3527
Seat booked!
Would you like to (a) book a seat, (b) see the amount of available seats
or (c) update a flight? a
Enter the flight number: 3000
Flight not found.
Would you like to (a) book a seat, (b) see the amount of available seats
or (c) update a flight? a
Enter the flight number: 3017
Seat booked!
Would you like to (a) book a seat, (b) see the amount of available seats
or (c) update a flight? b
Available seats on flight 3527 to Madrid (on-time): 179
Available seats on flight 3017 to Paris (delayed): 134
```