Object Oriented Programming – SCJ2153

Class and Object

Associate Prof. Dr. Norazah Yusof

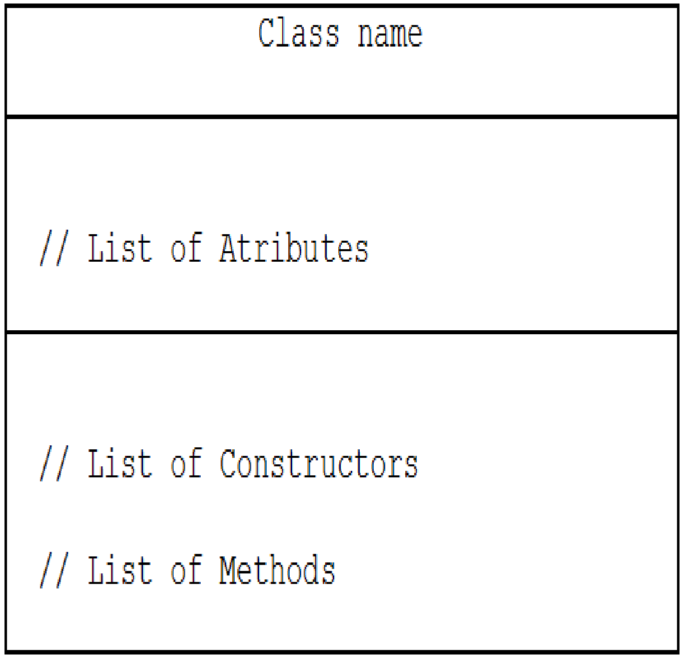
Classes

- Java program consists of classes.
- Class is a template for creating objects.
- Class normally consists of 3 components:
  1. data members/attributes (also known as fields),
  2. contructors (a special kind of method)
  3. methods

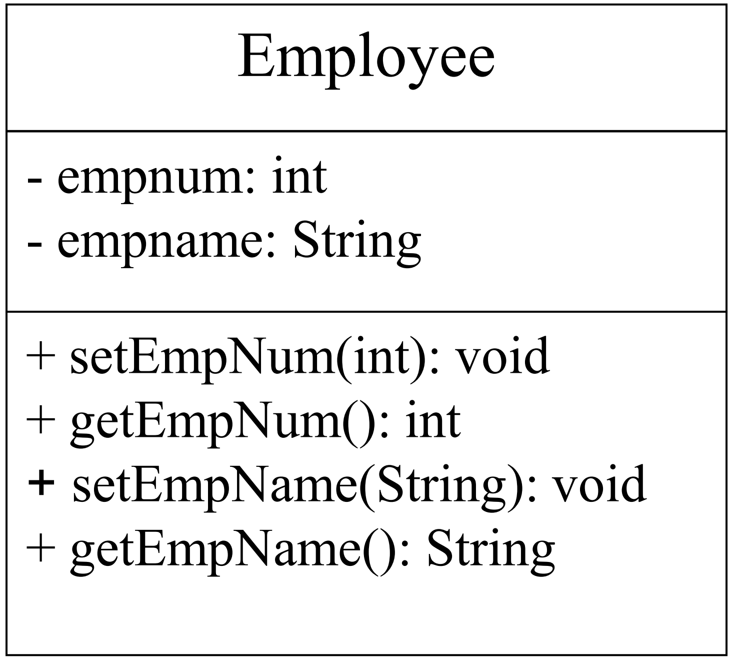
Class Diagram

- Before a class is defined, it is always helpful to design using a class diagram.
- Class diagram is a set of standard diagram that graphically shows the object oriented system.
- The diagram consists of a box that is divided into 3 parts:
  - Class name
  - Attributes
  - Constructors and methods

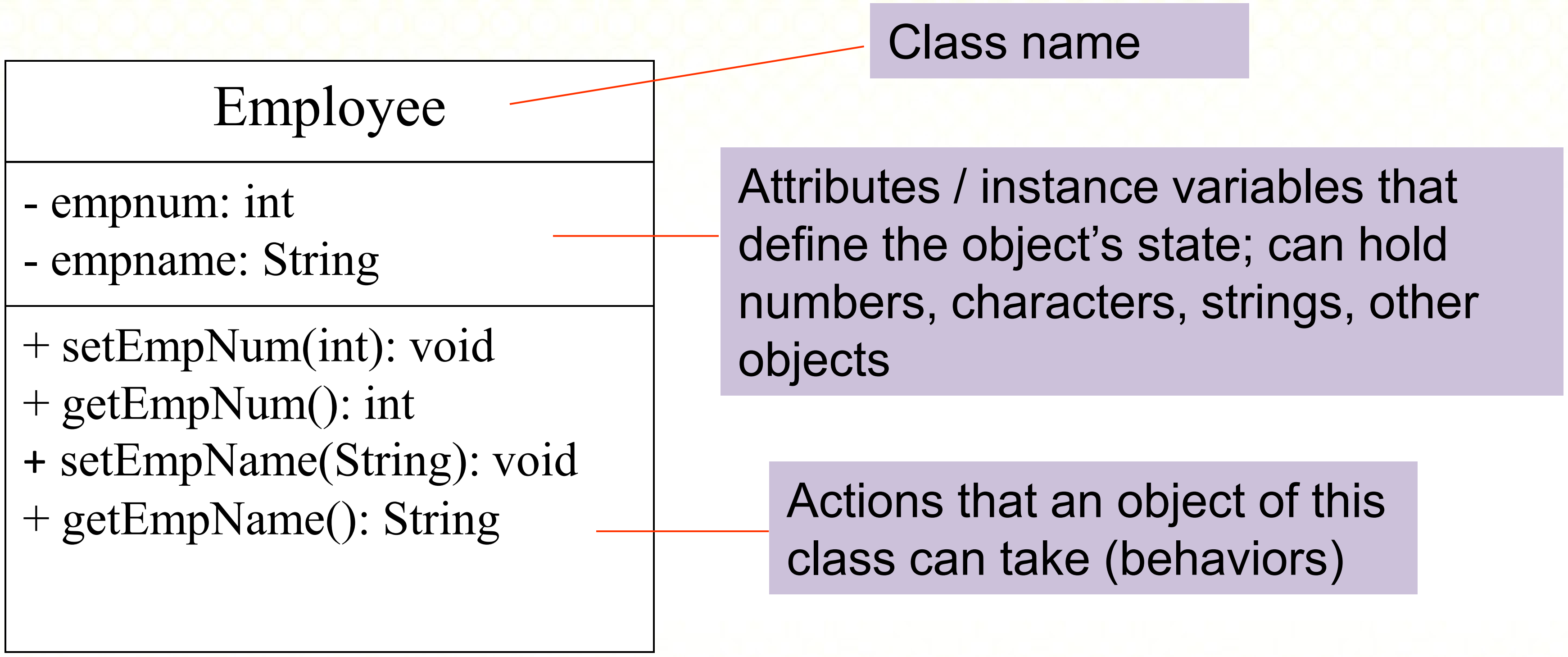
Structure of Class Diagram



Example of a Class Diagram



Class Diagram for Employee


Class name
Attributes / instance variables that define the object's state; can hold numbers, characters, strings, other objects
Actions that an object of this class can take (behaviors)

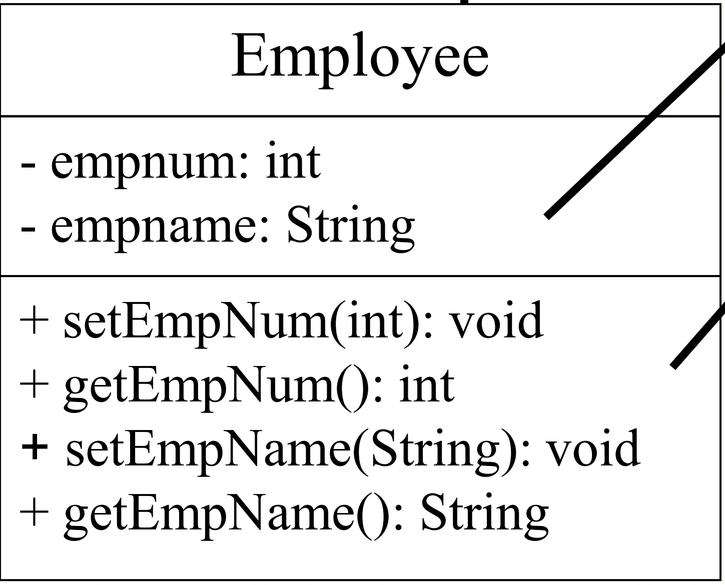
```
Employee
-----------------------------
- empnum: int
- empname: String
-----------------------------
+ setEmpNum(int): void
+ getEmpNum(): int
+ setEmpName(String): void
+ getEmpName(): String
```

The – symbol means private access modifier.
The + symbol means public access modifier.

Structure of a Class Definition

```
public class ClassName {
   // attributes definitions

   //constructor definitions
   public ClassName(paramList) {

   }

   //method definitions
    public returnType methodName(paramList) {

    }

}
```

Class Definition



class diagram

```
public class Employee {
  private int empnum;
  private String empname;

  public void setEmpNum(int num) {
   empnum = num;
   }

  public int getEmpNum() {
    return empnum;
   }

 public void setEmpName(String name) {
    empname = name;
  }

 public String getEmpName() {
    return empname;
  }
}
```

Information Hiding Principle Accessor and Mutator method

- Information hiding using encapsulation
  - Attributes are usually `private`
  - Client application accesses them only through `public` interfaces
- Mutator method
  - Controls data values used to set variable
- Accessor method
  - Controls how value retrieved

Accessors and Mutators

- For the Employee class example, the accessors and mutators are:
  - setEmpNum : Sets the value of the employee's number field.

    public void setEmpNum(int num) { }
  - setEmpName : Sets the value of the employee's name field.

    public void setEmpName(String name)
  - getEmpNum : Returns the value of the employee's number field.

    public int getEmpNum() { }
  - getEmpName : Returns the value of the employee's name field.

    public String getEmpName() { }

  Other names for these methods are *getters* and *setters*.

Declaring Object Reference Variable and Create Object

- To reference an object, assign the object to a **reference variable**.

- To declare a reference variable, use the syntax:

```
ClassName objectRefVar;
```

- To create object:

```
objectRefVar = new ClassName();
```

Example:

```
Employee emp1;
emp1 = Employee();
```

Declaring/Creating Objects in a Single Step

```
ClassName objectRefVar = new ClassName();
```

Assign object reference

Create an object

Example:

```
Employee emp1 = new Employee();
```

Accessing Objects

- After object instantiated, methods accessed using:
    - Object’s identifier
    - Dot
    - Data/Method call
- Referencing the object’s data:

```
objectRefVar.data
```

- Example:

```
emp1.empnum
```

- Invoking the object’s method:

```
objectRefVar.methodName(arguments)
```

- Example:

```
emp1.getEmpNum()
```

Constructors

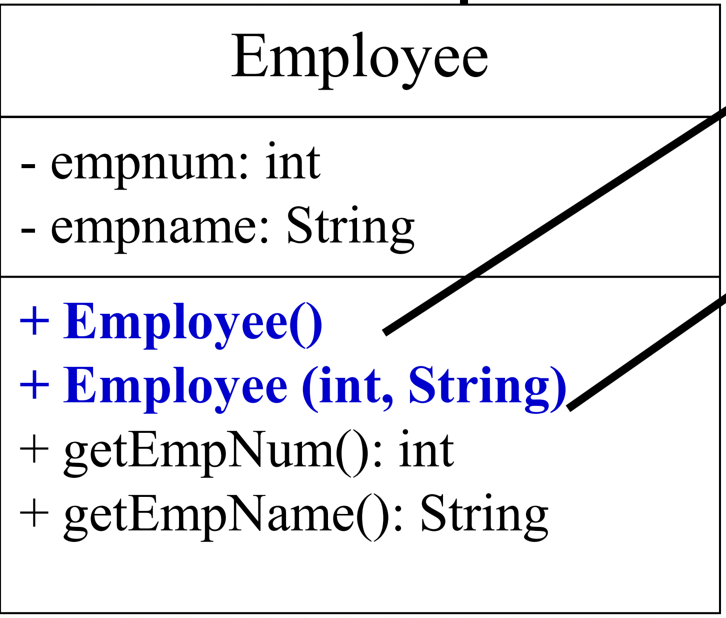
- Special kind of methods for creating objects of a class, which play the role of **initializing** objects.
- The name of constructor must be the same as name of class.
- Constructors do not have a return type and they do not return a value.
- If a programmer does not define any constructors in a class, Java provides one default constructor i.e. a no argument constructor.

Constructors

- Constructors may take parameters.
  - If a constructor has one parameter, it is called a **one-argument constructor**.
- If a class has more than one constructors, they must have different numbers and/or types of parameters.
  - This is called **constructor overloading**
- Syntax:

```
public ClassName(paramList) {


}
```

Class Definition with constructor



class diagram

```
public class Employee {
  private int empnum;
  private String empname;

  public Employee() {
    System.out.println (“Start Employee”);
  }
  public Employee(int num, String name) {
    empnum = num;
    empname = name;
  }

  public int getEmpNum() {
    return empnum;
   }

  public String getEmpName() {
    return empname;
  }
}
```

Creating Object == invoking Constructors

- Constructors are invoked using the **new operator** when an object is created. Constructors play the role of **initializing** objects.
- Syntax:

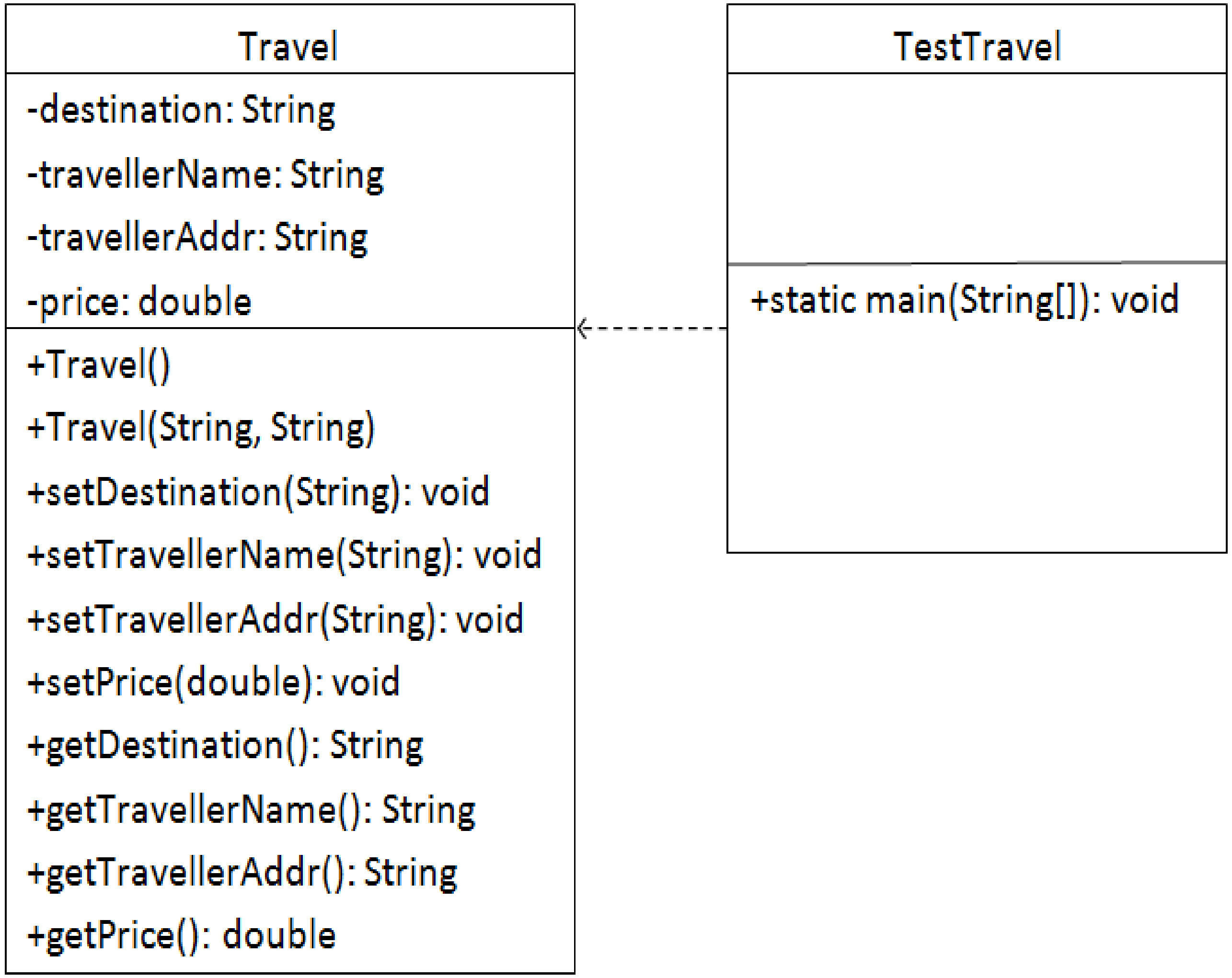
```
new ClassName();
```

Example:

```
new Employee();

new Employee(123, “Ahmad”);
```

Programming Example

- Given the following class diagram about `Travel` write complete Java program for the traveller.


Travel
-destination: String
-travellerName: String
-travellerAddr: String
-price: double
+Travel()
+Travel(String, String)
+setDestination(String): void
+setTravellerName(String): void
+setTravellerAddr(String): void
+setPrice(double): void
+getDestination(): String
+getTravellerName(): String
+getTravellerAddr(): String
+getPrice(): double
TestTravel
+static main(String[]): void

Travel.java

```java
public class Travel {
  private String destination, name, address;
  private double price;

  public Travel(){

  }

  public Travel(String name, String destination){
    this.name = name ;
    this.destination = destination;
  }

  public String getDestination(){
    return destination;
  }
```

Travel.java (cont.)

```java
public String getName(){
      return name;
   }

   public String getAddress(){
      return address;
   }

   public double getPrice(){
      return price;
   }

   public void setDestination(String d){
      destination = d;
   }

   public void setName(String n){
      name = n;
   }
```

Travel.java (cont.)

```
  public void setAddress(String a){
    address = a;
  }

  public void setPrice(double p){
    price = p;
  }

 public void display(){
    System.out.println("\nDestination = "+destination);
    System.out.println("Name of traveller = "+name);
    System.out.println("Address of traveller = " +
     address);
    System.out.printf("Price = RM %.2f \n", price);
  }
}
```

TestTravel.java

```
public class TestTravel {
  public static void main(String[] args)   {
     Travel ob1 = new Travel();
     Travel ob2 = new Travel("Eyan","Indonesia");
     ob1.setName("Toh");
     ob1.setDestination("Italy");
     ob1.setAddress("NO.1 Tmn Emas,30220 Ipoh,
        Perak,Malaysia");
     ob1.setPrice(5999);
     ob1.display();
     ob2.setAddress("N0.100 Tmn Raja,32223 Taiping,
        Malaysia");
     ob2.setPrice(3999);
     ob2.display();
   }
}
```