Building Blocks

Objects

Creating objects

- object is an instance of the class *(house → blueprint)*
- new object is created using a keyword `new` :

```
Student s = new Student()
```

- when object is created, *the constructor* of the object is called

```java
// in file Student.java
public class Student {
  public Student() {
    System.out.println("New student is created.");
  }
}

// in file MyApp.java
public class MyApp {
  public static void main(String[] args) {
    Student s = new Student();
  }
}
```

the constructor

(like method, but no return type)

```
$ java MyApp

New student is created.
```

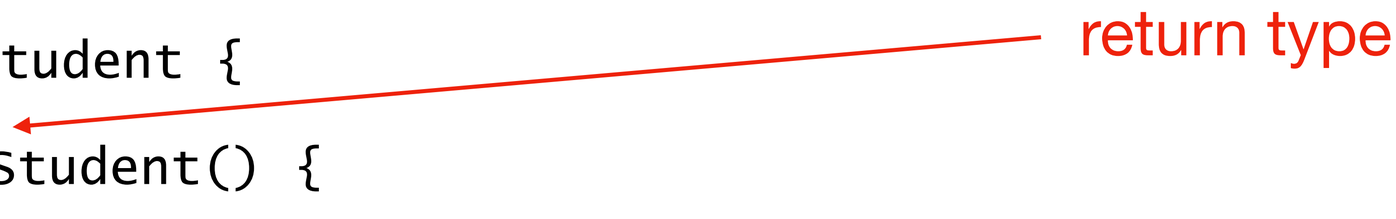
new object is created

and constructor is called

```
// will this compile?
public class Student {
  public void Student() {
    System.out.println("New student is created.");
  }
}

// YES!
// But here Student() is just a method, not a constructor
// (it will not be called when new object is created)

// good practice is to write methods with lowercase first letter
// (but exam creators like these kind of practical jokes)
```



```
// if you don't provide any constructor, the compiler will generate
// simple no-argument constructor: public Student() { }


// reading and modifying fields:
public class Student {
  String name;    // instance variable


  public static void main(String[] args) {
    Student s = new Student();     // creating an object
    s.name = "John Wayne";         // set variable
    System.out.println(s.name);    // get variable
  }
}
```

Order of initialization

- the code between two brackets `{...}` is called **code block**
- **instance initializer** - code block outside the method
- order of initialization:
    1. fields and instance initializer blocks in order in which they appear
    2. constructor runs in the end

```
public class Dog {

  private String name = "Chip";

  public Dog() {
    name = "Teddy";
    System.out.println("Inside the constructor...");
  }

  { System.out.println("Inside the initializer block..."); }


  public static void main(String[] args) {
    Dog dog = new Dog();
    System.out.println(dog.name);
  }

}
```

2

1

3

```
$ java Dog

Inside the initializer block...
Inside the constructor...
Teddy
```