4: Initialization

12/01/2003

Initialization

(modified)

developmentor

Instance field default values

- **Instance fields set to default values when object created**
  - `0` for numeric types
  - `false` for `bool`
  - `'\x0000'` for `char`
  - `null` for references

instance fields

```
class Rational
{
  int numerator;
  int denominator;
  ...
}
```

set to default values

```
Rational a = new Rational();
```

```
a
numerator    0
denominator  0
```

Variable initializer

- **Instance fields can be initialized at point of definition**
  - called *variable initializer*
  - executed each time an object is created
  - convenient way to overwrite default values

initialize

```
class Rational
{
  int numerator;
  int denominator = 1;

  ...
}
```

Constructor

- **Class can supply *constructor* to do initialization**
  - automatically invoked when object created
  - implemented using same name as class with no return type

constructor

```
class Rational
{
  int numerator;
  int denominator;

  public Rational(int numerator, int denominator)
  {
    this.numerator   = numerator;
    this.denominator = denominator;
  }
  ...
}
```

Invoking constructor

- **Constructor automatically invoked when object created**

create object,
invoke constructor

```
Rational a = new Rational(2, 5);
```

```
a
numerator    2
denominator  5
```

Multiple constructors

- **Class can supply multiple constructors**
  - parameter lists must be different

constructor

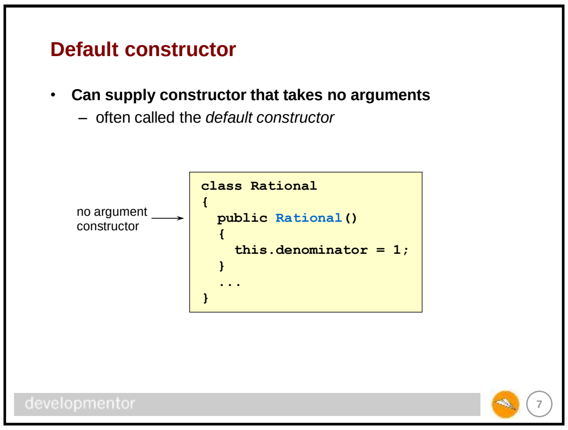
constructor

```
class Rational
{
  public Rational(int numerator, int denominator)
  {
    this.numerator   = numerator;
    this.denominator = denominator;
  }

  public Rational(int numerator)
  {
    this.numerator   = numerator;
    this.denominator = 1;
  }
  ...
}
```

4: Initialization

12/01/2003



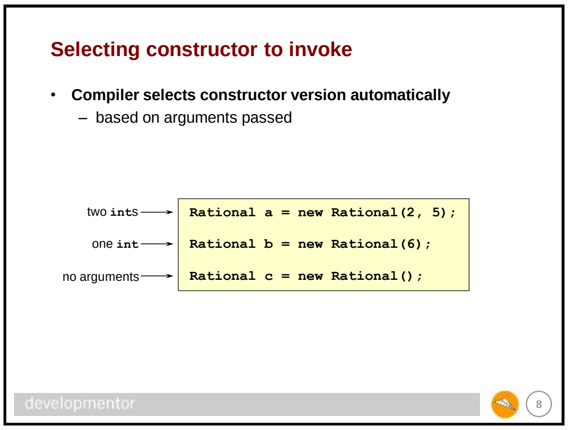
Default constructor

- **Can supply constructor that takes no arguments**
  - often called the *default constructor*

no argument constructor

```
class Rational
{
  public Rational()
  {
    this.denominator = 1;
  }
  ...
}
```

developmentor 7



Selecting constructor to invoke

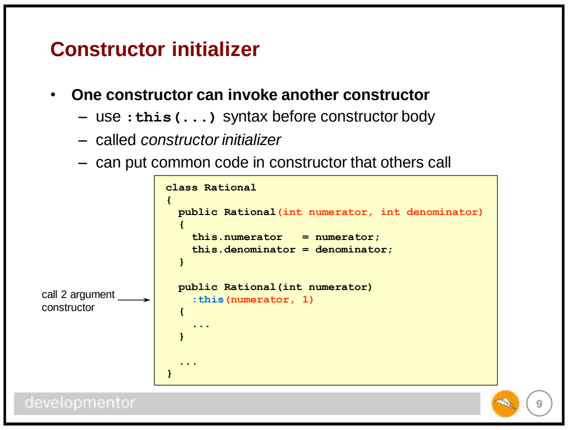
- **Compiler selects constructor version automatically**
  - based on arguments passed

two ints → `Rational a = new Rational(2, 5);`

one int → `Rational b = new Rational(6);`

no arguments → `Rational c = new Rational();`

developmentor 8



Constructor initializer

- **One constructor can invoke another constructor**
  - use `:this(...)` syntax before constructor body
  - called *constructor initializer*
  - can put common code in constructor that others call

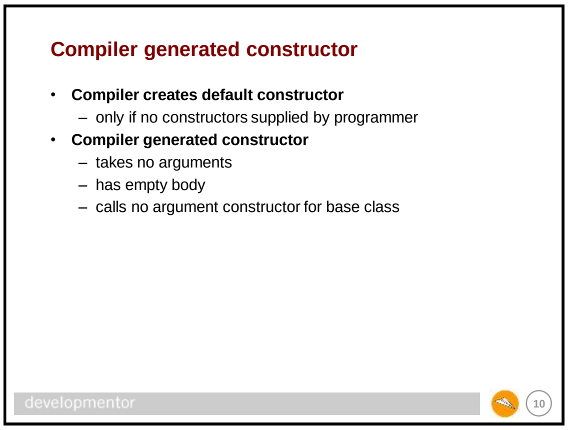
call 2 argument constructor

```
class Rational
{
  public Rational(int numerator, int denominator)
  {
    this.numerator   = numerator;
    this.denominator = denominator;
  }

  public Rational(int numerator)
    :this(numerator, 1)
  {
    ...
  }

  ...
}
```

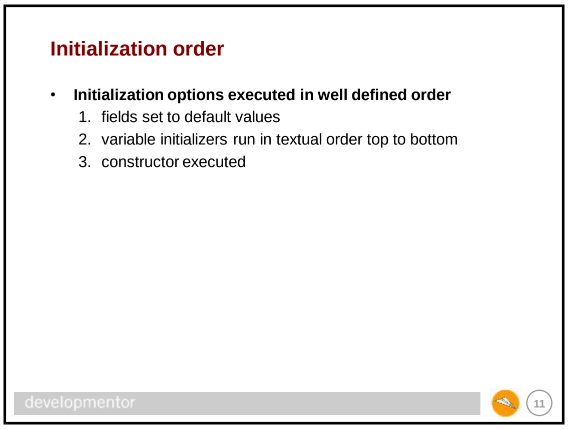
developmentor 9



Compiler generated constructor

- **Compiler creates default constructor**
  - only if no constructors supplied by programmer
- **Compiler generated constructor**
  - takes no arguments
  - has empty body
  - calls no argument constructor for base class

developmentor 10



Initialization order

- **Initialization options executed in well defined order**
  1. fields set to default values
  2. variable initializers run in textual order top to bottom
  3. constructor executed

developmentor 11