Aula 3

Encapsulamento e Funções de acesso

Encapsulamento

Encapsulation

Information Hiding

- Porque fazer as variáveis membro “private” ?
  Resposta usando a “analogia do controle remoto”
- Na programação orientada a objetos, encapsulamento (também chamado de ocultamento de informações) é o processo de manter os detalhes sobre como um objeto é implementado escondido dos usuários desse objeto.
- Os usuários do objeto acessam-no através de uma interface pública. Desta forma, os usuários são capazes de usar o objeto sem ter que entender como ele é implementado.

Encapsulamento (cont.)

- Em C ++, vamos implementar encapsulamento através de especificadores de acesso.
- Normalmente:
  - Todas as variáveis de membro da classe são feitas “private”(esconde os detalhes de implementação), e
  - a maioria das funções de membro são tornados “public” (expondo uma interface para o utilizador).

Encapsulamento (cont.)

- Beneficios:
  - Classes encapsuladas são mais fáceis de usar e reduzem a complexidade dos programas (reduz erros)
  - Classes encapsuladas ajudam a proteger os seus dados e evitam o uso indevido

Encapsulamento (cont.)

- Benefícios:
  - Classes encapsuladas são mais fáceis de usar e reduzem a complexidade dos programas (reduz erros)
  - Classes encapsuladas ajudam a proteger os seus dados e evitam o uso indevido

```
1  class MyString
2  {
3      char *m_string; // we'll dynamically allocate our string here
4      int m_length; // we need to keep track of the string length
5  };
```

Encapsulamento (cont.)

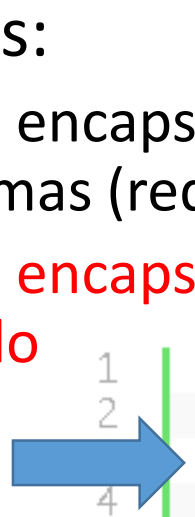
- Benefícios:
  - Classes encapsuladas são mais fáceis de usar e reduzem a complexidade dos programas (reduz erros)
  - Classes encapsuladas ajudam a proteger os seus dados e evitam o uso indevido

```
1  class IntArray
2  {
3  public:
4      int m_array[10];
5  };
```

```
1  int main()
2  {
3      IntArray array;
4      array.m_array[16] = 2; // invalid array index, now we overwrote memory tha
   t we don't own
5  }
```

Encapsulamento (cont.)

- Benefícios:
  - Classes encapsuladas são mais fáceis de usar e reduzem a complexidade dos programas (reduz erros)
  - Classes encapsuladas ajudam a proteger os seus dados e evitam o uso indevido



```
1  class IntArray
2  {
3  private:
4      int m_array[10]; // user can not access this directly any more
5
6  public:
7      void setValue(int index, int value)
8      {
9          // If the index is invalid, do nothing
10         if (index < 0 || index >= 10)
11             return;
12
13         m_array[index] = value;
14     }
15 };
```

Encapsulamento (cont.)

- Benefícios:
  - Classes encapsuladas são mais fá programas (reduz erros);
  - Classes encapsuladas ajudam a p indevido;
  - Classes encapsulados são mais fáceis de mudar.

```
1  #include <iostream>
2
3  class Something
4  {
5  public:
6      int m_value1;
7      int m_value2;
8      int m_value3;
9  };
10
11 int main()
12 {
13     Something something;
14     something.m_value1 = 5;
15     std::cout << something.m_value1 << '\n';
16 }
```

Encapsulamento (cont.)

```cpp
#include <iostream>

class Something
{
private:
    int m_value[3]; // note: we changed the implementa

public:
    // We have to update any member functions to refle
on
    void setValue1(int value) { m_value[0] = value; }
    int getValue1() { return m_value[0]; }
};

int main()
{
    // But our program still works just fine!
    Something something;
    something.setValue1(5);
    std::cout << something.getValue1() << '\n';
}
```

```cpp
#include <iostream>

class Something
{
private:
    int m_value1;
    int m_value2;
    int m_value3;

public:
    void setValue1(int value) { m_value1 = value; }
    int getValue1() { return m_value1; }
};

int main()
{
    Something something;
    something.setValue1(5);
    std::cout << something.getValue1() << '\n';
}
```

Encapsulamento (cont.)

- Benefícios:
  - Classes encapsuladas são mais fáceis de usar e reduzem a complexidade dos programas (reduz erros);
  - Classes encapsuladas ajudam a proteger os seus dados e evitam o uso indevido;
  - Classes encapsulados são mais fáceis de mudar;
  - Debug de Classes encapsuladas são mais simples.

Funções de acesso

Uma função de acesso é uma pequena função pública que fornece ou altera o valor de uma variável membro privada.
Por exemplo, em uma classe String:

```
1  class MyString
2  {
3  private:
4      char *m_string; // we'll dynamically allocate our string here
5      int m_length; // we need to keep track of the string length
6
7  public:
8      int getLength() { return m_length; } // access function to get value of m_
   length
9  };
```

Other sizes

Funções de acesso normalmente vêm em dois formas: getters e setters.
Getters são funções que retornam o valor de uma variável membro privada.
Setters são funções que definem o valor de uma variável membro privada.

Funções de acesso

```
class Date
{
private:
    int m_month;
    int m_day;
    int m_year;

public:
    int getMonth() { return m_month; } // getter for month
    void setMonth(int month) { m_month = month; } // setter for month

    int getDay() { return m_day; } // getter for day
    void setDay(int day) { m_day = day; } // setter for day

    int getYear() { return m_year; } // getter for year
    void setYear(int year) { m_year = year; } // setter for year
};
```

Constructors and Destructors

Tarefa de Casa :- Estudar os tópicos e implementar exemplos (C++, Java, Python, etc... Livre !!!!) valorizando o entendimento dos seguintes conceitos:

- O que são?
- Para que servem ?
- Quando são utilizados ?
- Como são utilizados ?
- Quando é necessario e quando não é necessario usar Destructors ?

Montagem de um diagrama de classes

Este sistema será utilizado para gerenciar e controlar os processos de produção em uma fábrica das Industrias G. A fábrica faz vários tipos de dispositivos mecânicos. Ela tem 10 linhas de montagem, cada uma das linhas pode ser utilizada para fabricar qualquer um dos seus produtos. Uma linha de montagem é atribuída a um produto por um período fixo de tempo (em qualquer lugar de algumas horas a alguns dias) - isso é chamado de uma corrida de produção (product run). Durante corrida de produção, a linha de montagem faz um número especifico de unidades do produto.

Cada produto é montado em vários etapas. Enquanto os produtos sendo montados se movem ao longo da linha de montagem, por sua vez, eles são trabalhados por uma série de robôs. Cada robô realiza uma etapa antes do produto se deslocar para a próximo etapa (e um robô diferente). Cada robô é dedicado a apenas uma etapa de fabricação.

Cada produto é composto por peças. As peças podem ser compradas de fornecedores, ou podem ser compostas de produtos menores que foram construídos na fabrica (em produtos corridas anteriores). Em cada etapa de fabricação, um dado subconjunto destas partes é montado. Peças esperando para a montagem são mantidos em caixas numeradas; os robôs sabem qual caixas devem acessar para pegar as peças necessárias.

Para cada conjunto completo é fornecido um número de série. Quando os pedidos de produtos estão completos, os números de série dos produtos vendidos são registrados junto com o pedido.

Montagem de um diagrama de classes

Este sistema será utilizado para gerenciar e controlar os processos de produção em uma fábrica das Industrias G. A fábrica faz vários tipos de dispositivos mecânicos. Ela tem 10 linhas de montagem, cada uma das linhas pode ser utilizada para fabricar qualquer um dos seus produtos. Uma linha de montagem é atribuída a um produto por um período de tempo fixo (em qualquer lugar de algumas horas a alguns dias) - isso é chamado de uma corrida de produção (product run). Durante corrida de produção, a linha de montagem faz um número específico de unidades do produto.

Cada produto é montado em vários etapas. Enquanto os produtos sendo montados se movem ao longo da linha de montagem, por sua vez, eles são trabalhados por uma série de robôs. Cada robô realiza uma etapa antes do produto se deslocar para a próximo etapa (e um robô diferente). Cada robô é dedicado a apenas uma etapa de fabricação.

Cada produto é composto por peças. As peças podem ser compradas de fornecedores, ou podem ser compostas de produtos menores que foram construídos na fábrica (em produtos corridas anteriores). Em cada etapa de fabricação, um dado subconjunto destas partes é montado. Peças esperando para a montagem são mantidos em caixas numeradas; os robôs sabem qual caixas devem acessar para pegar as peças necessárias.

Para cada conjunto completo é fornecido um número de série. Quando os pedidos de produtos estão completos, os números de série dos produtos vendidos são registrados junto com o pedido.

- Processos de produção
- Fábrica
- Dispositivos mecânicos
- Linha de montagem
- **Produto**
- **Corrida de produção**
- Unidade de produto
- **Etapa**
- **Robo**
- Peça
- **Fornecedor**
- **Caixas numeradas (Bin)**
- Montagem
- **Pedido**

Processos -> Metodos que alteram a classe , ou o objeto

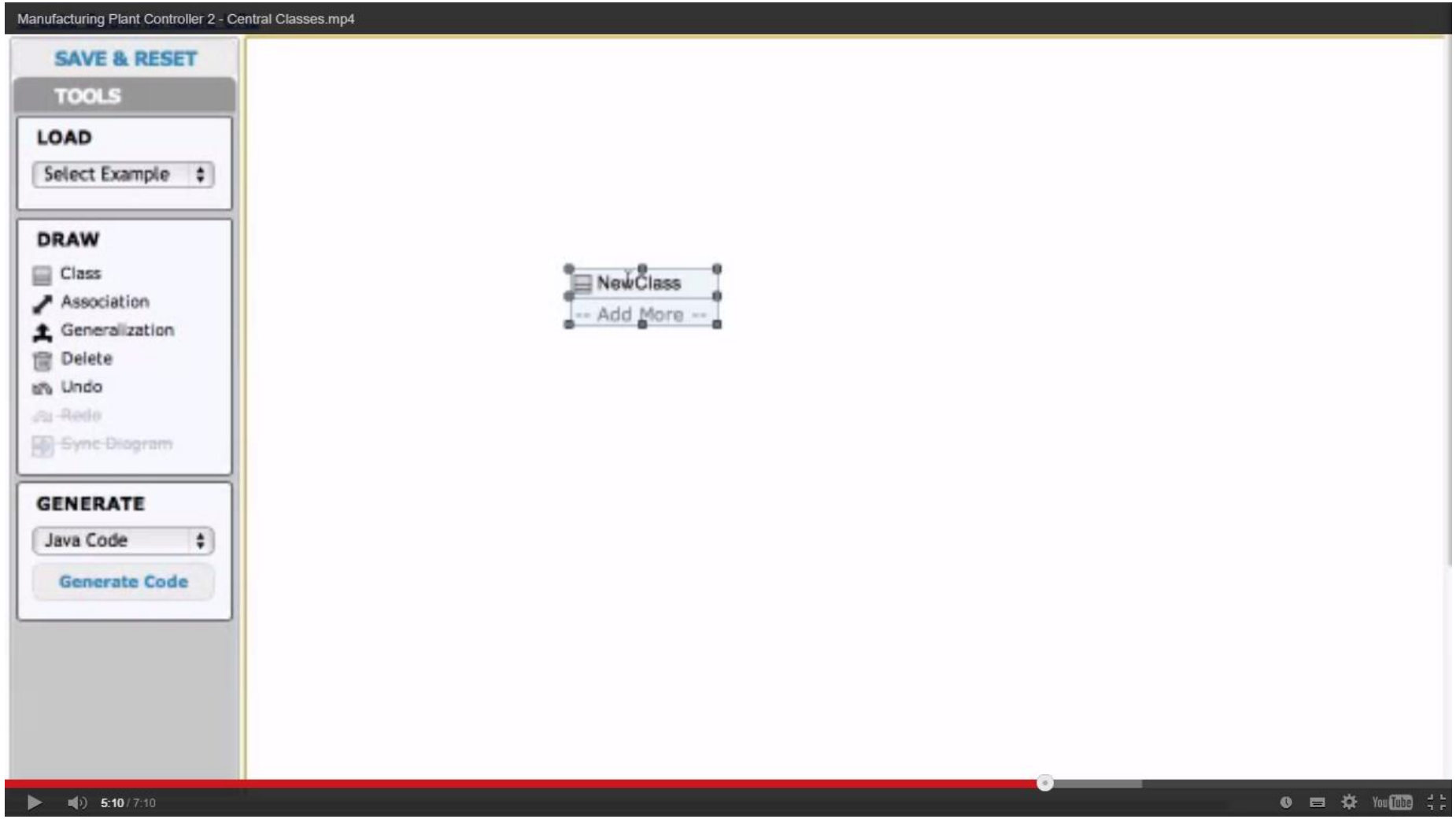
Singleton

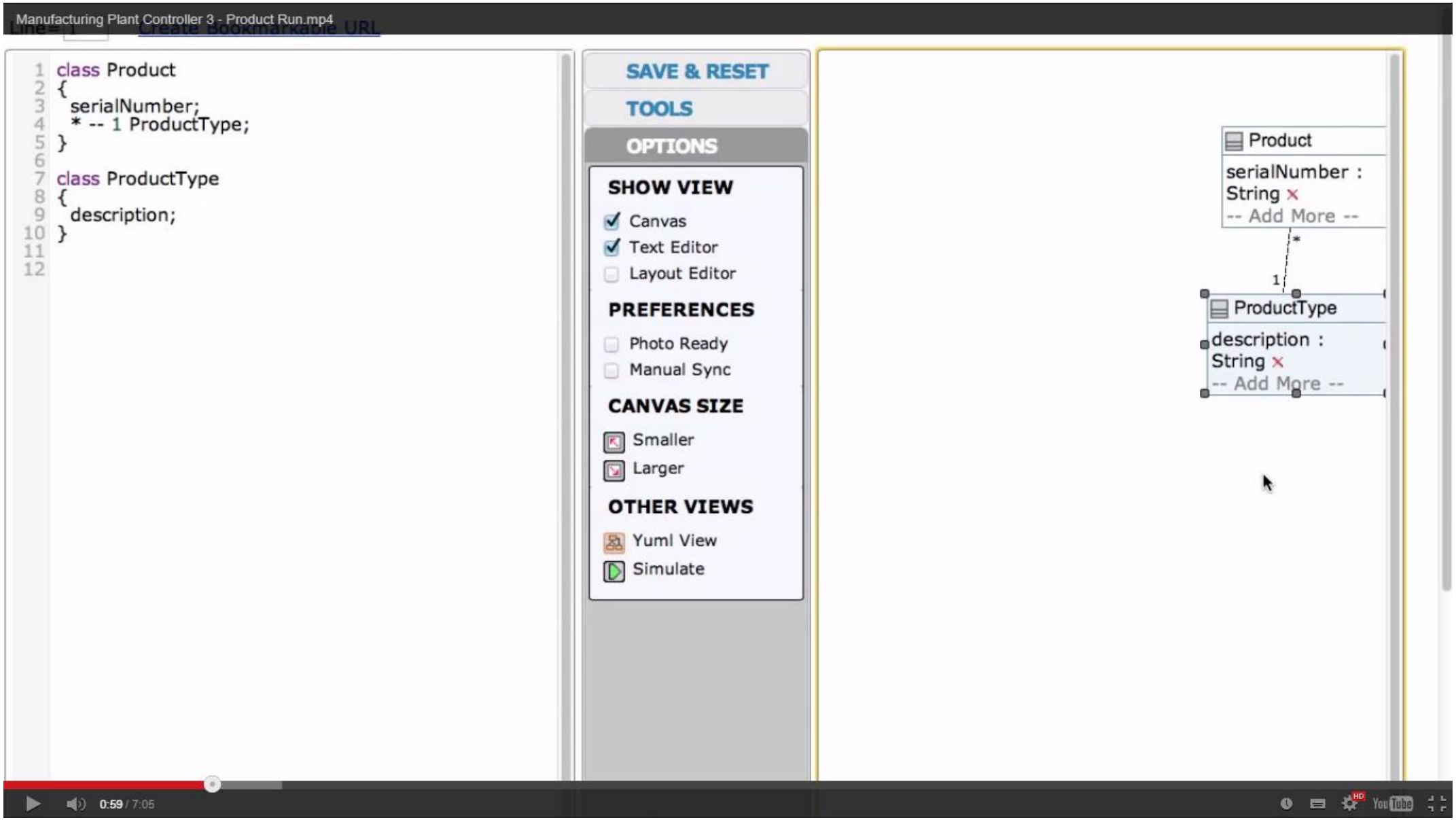
? = Produto

-> TipoDeProduto -> atributo = descrição

-> Produto -> atributo = numero de série

Editores UML




Manufacturing Plant Controller 3 - Product Run.mp4
Create bookmarkable URL
class Product
{
serialNumber;
* -- 1 ProductType;
}
class ProductType
{
description;
}
SAVE & RESET
TOOLS
OPTIONS
SHOW VIEW
Canvas
Text Editor
Layout Editor
PREFERENCES
Photo Ready
Manual Sync
CANVAS SIZE
Smaller
Larger
OTHER VIEWS
Yuml View
Simulate
Product
serialNumber : String
-- Add More --
*
1
ProductType
description : String
-- Add More --
0:59 / 7:05

Este sistema será utilizado para gerenciar e controlar os processos de produção em uma fábrica das Industrias G. A fábrica faz vários tipos de dispositivos mecânicos. Ela tem 10 linhas de montagem, cada uma das linhas pode ser utilizada para fabricar qualquer um dos seus produtos. Uma linha de montagem é atribuída a um produto por um período de tempo fixo (em qualquer lugar de algumas horas a alguns dias) - isso é chamado de uma corrida de produção (product run). Durante corrida de produção, a linha de montagem faz um número específico de unidades do produto.

Cada produto é montado em vários etapas. Enquanto os produtos sendo montados se movem ao longo da linha de montagem, por sua vez, eles são trabalhados por uma série de robôs. Cada robô realiza uma etapa antes do produto se deslocar para a próximo etapa (e um robô diferente). Cada robô é dedicado a apenas uma etapa de fabricação.

Cada produto é composto por peças. As peças podem ser compradas de fornecedores, ou podem ser compostas de produtos menores que foram construídos na fábrica (em produtos corridas anteriores). Em cada etapa de fabricação, um dado subconjunto destas partes é montado. Peças esperando para a montagem são mantidos em caixas numeradas; os robôs sabem qual caixas devem acessar para pegar as peças necessárias.

Para cada conjunto completo é fornecido um número de série. Quando os pedidos de produtos estão completos, os números de série dos produtos vendidos são registrados junto com o pedido.

Este sistema será utilizado para gerenciar e controlar os processos de produção em uma fábrica
Industrias G. A fábrica faz vários tipos de dispositivos mecânicos. Ela tem 10 linhas de montagem, cada uma das linhas pode ser utilizada para fabricar qualquer um dos seus produtos. Uma linha de montagem é atribuída a um produto por um período de tempo fixo (em qualquer lugar de algumas horas a alguns dias) - isso é chamado de uma corrida de produção (product run). Durante corrida de produção, a linha de montagem faz um número específico de unidades do produto.

Manufacturing Plant Controller 3 - Product Run.mp4

Line = 1 Create Bookmarkable URL

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5  }
6
7  class ProductType
8  {
9    description;
10 }
11
12 class AssemblyLine
13 {
14   Integer number;
15 }
16
17 class NewClass
18 {
19 }
20
21
```

SAVE & RESET

TOOLS

LOAD

Select Example

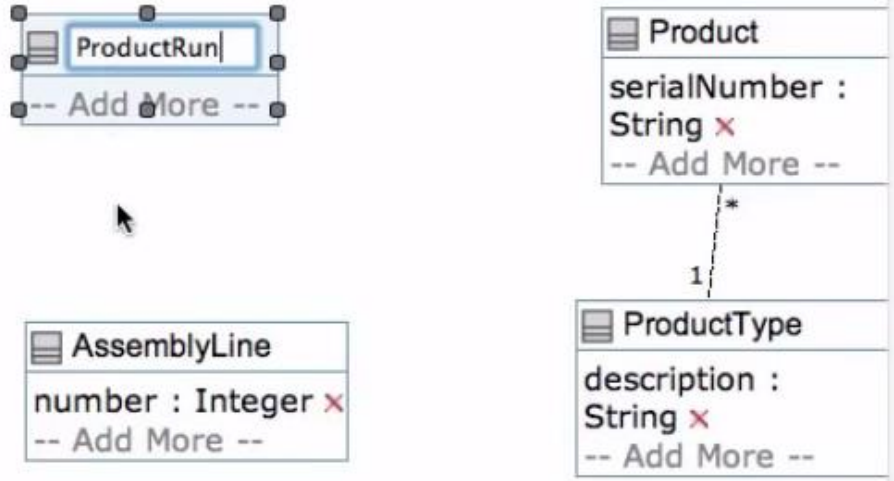
DRAW

- Class
- Association
- Generalization
- Delete
- Undo
- Redo
- Sync Diagram

GENERATE

Java Code

Generate Code



4:17 / 7:05

Manufacturing Plant Controller 3 - Product Run.mp4

Line = 1 Create Bookmarkable URL

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5  }
6
7  class ProductType
8  {
9    description;
10 }
11
12 class AssemblyLine
13 {
14   Integer number;
15   1 -- * ProductRun;
16 }
17
18 class ProductRun
19 {
20   timePeriod;
21   * -- 1 ProductType;
22 }
23
24
```

SAVE & RESET

TOOLS

LOAD

Select Example

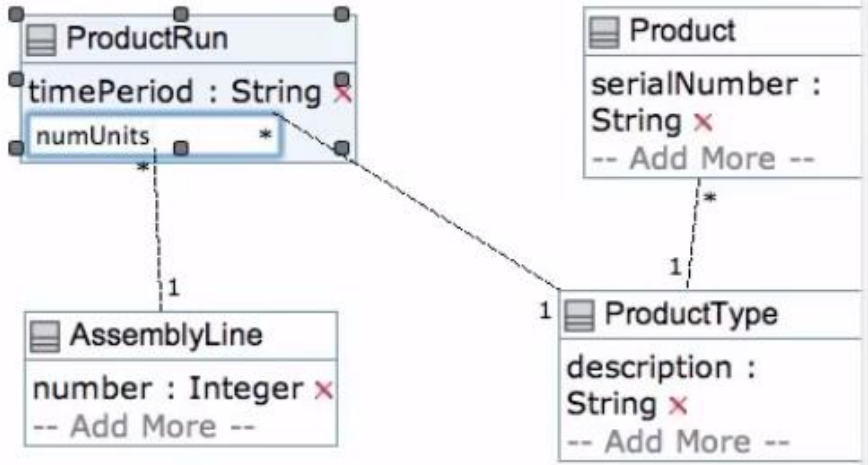
DRAW

- Class
- Association
- Generalization
- Delete
- Undo
- Redo
- Sync Diagram

GENERATE

Java Code

Generate Code



6:27 / 7:05

Manufacturing Plant Controller 4 - Assembly Step.mp4

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6  }
7
8  class ProductType
9  {
10   description;
11 }
12
13 class AssemblyLine
14 {
15   Integer number;
16   1 -- * ProductRun;
17 }
18
19 class ProductRun
20 {
21   timePeriod;
22   Integer numUnits;
23   * -- 1 ProductType;
24 }
25
26
```

SAVE & RESET

TOOLS

LOAD

Select Example

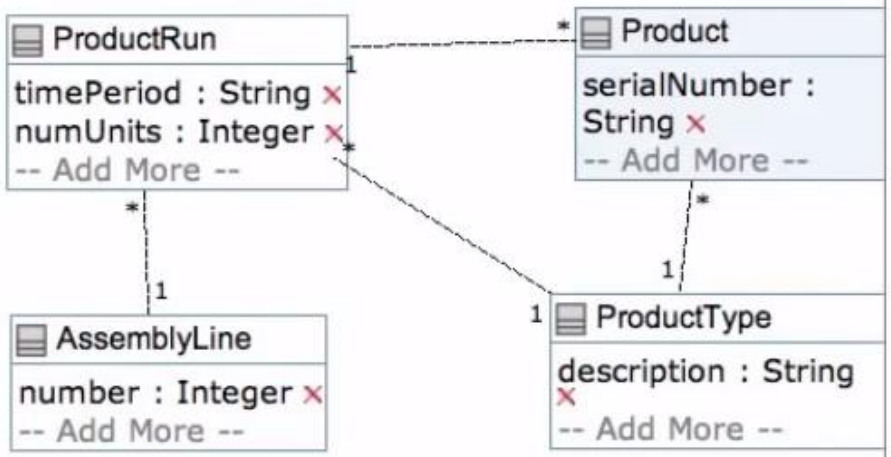
DRAW

- Class
- Association
- Generalization
- Delete
- Undo
- Redo
- Sync Diagram

GENERATE

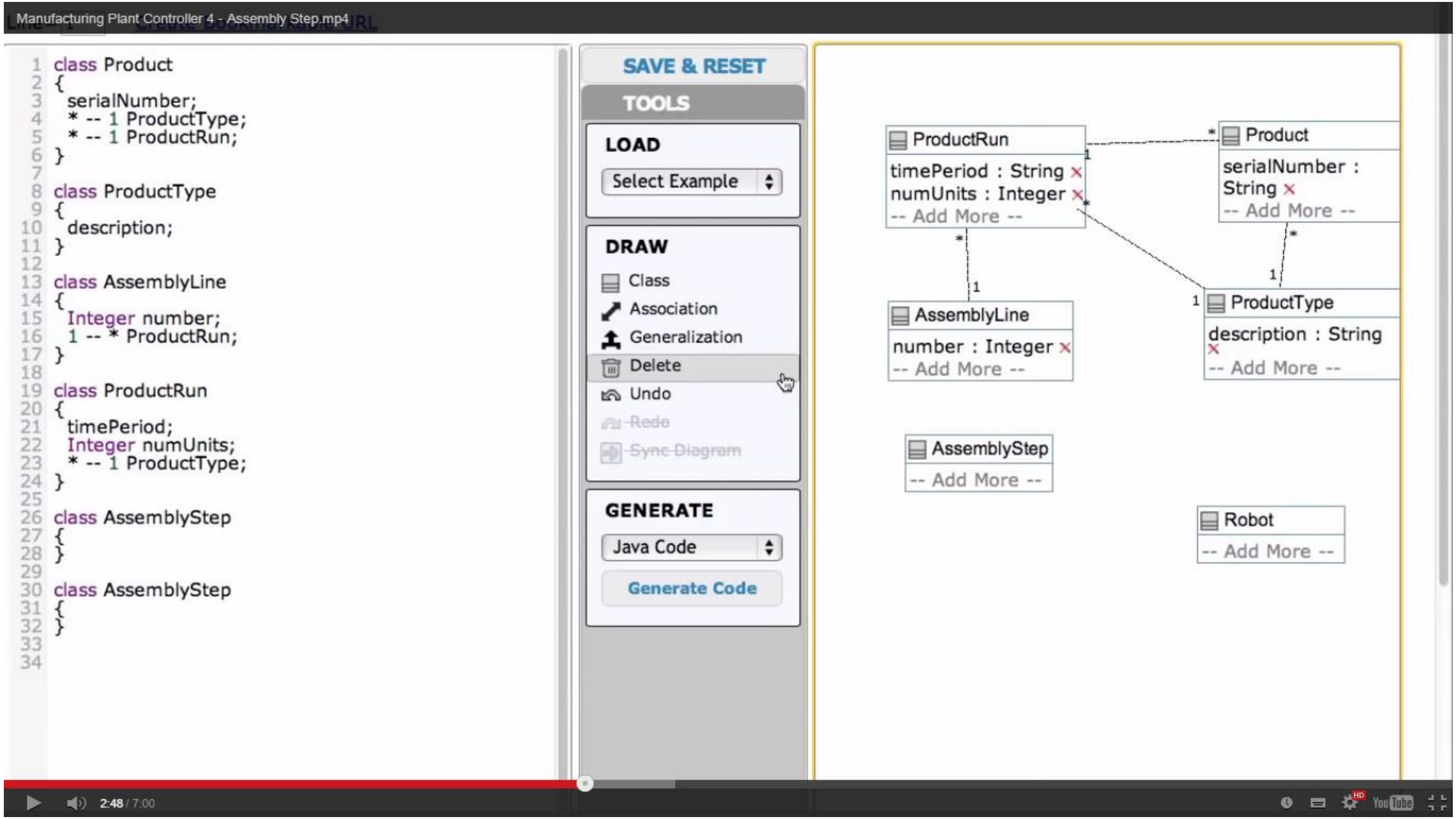
Java Code

Generate Code



1:24 / 7:00

Cada produto é montado em vários etapas. Enquanto os produtos são montados, eles se movem ao longo da linha de montagem, por sua vez, eles são trabalhados por uma série de robôs. Cada robô realiza uma etapa antes do produto se deslocar para a próximo etapa (e um robô diferente). Cada robô é dedicado a apenas uma etapa de fabricação.


Manufacturing Plant Controller 4 - Assembly Step.mp4
class Product
{
serialNumber;
* -- 1 ProductType;
* -- 1 ProductRun;
}
class ProductType
{
description;
}
class AssemblyLine
{
Integer number;
1 -- * ProductRun;
}
class ProductRun
{
timePeriod;
Integer numUnits;
* -- 1 ProductType;
}
class AssemblyStep
{
}
class AssemblyStep
{
}
SAVE & RESET
TOOLS
LOAD
Select Example
DRAW
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram
GENERATE
Java Code
Generate Code
ProductRun
timePeriod : String
numUnits : Integer
-- Add More --
Product
serialNumber : String
-- Add More --
AssemblyLine
number : Integer
-- Add More --
ProductType
description : String
-- Add More --
AssemblyStep
-- Add More --
Robot
-- Add More --
2:48 / 7:00

Manufacturing Plant Controller 4 - Assembly Step.mp4

Draw on the right, write (Umple) model code on the left. Generate Java, PHP or Ruby code from your models.
Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
class ProductType
{
  description;
}

class AssemblyLine
{
  Integer number;
  1 -- * ProductRun;
}

class ProductRun
{
  timePeriod;
  Integer numUnits;
  * -- 1 ProductType;
}

class AssemblyStep
{
}

class AssemblyStep
{
   * -- 1 ProductType;
}

class Robot
{
}

class RobotAllocation
{
}
```

SAVE & RESET

TOOLS

LOAD

Select Example

DRAW

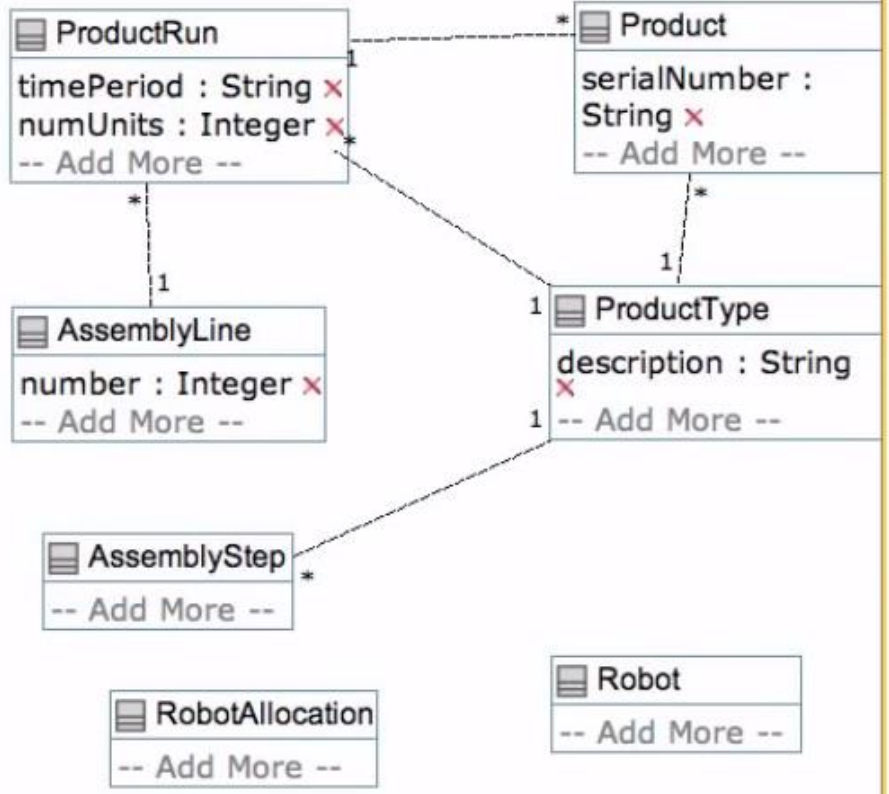
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code

OPTIONS



5:16 / 7:00

Manufacturing Plant Controller 4 - Assembly Step.mp4

Draw on the right, write (Umple) model code on the left. Generate Java, PHP or Ruby code from your models.

Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6  }
7
8  class ProductType
9  {
10   description;
11 }
12
13 class AssemblyLine
14 {
15   Integer number;
16   1 -- * ProductRun;
17 }
18
19 class ProductRun
20 {
21   timePeriod;
22   Integer numUnits;
23   * -- 1 ProductType;
24 }
25
26 class AssemblyStep
27 {
28   description;
29 }
30
31 class AssemblyStep
32 {
33     * -- 1 ProductType;
34 }
35
36 class Robot
```

SAVE & RESET

TOOLS

LOAD

Select Example

DRAW

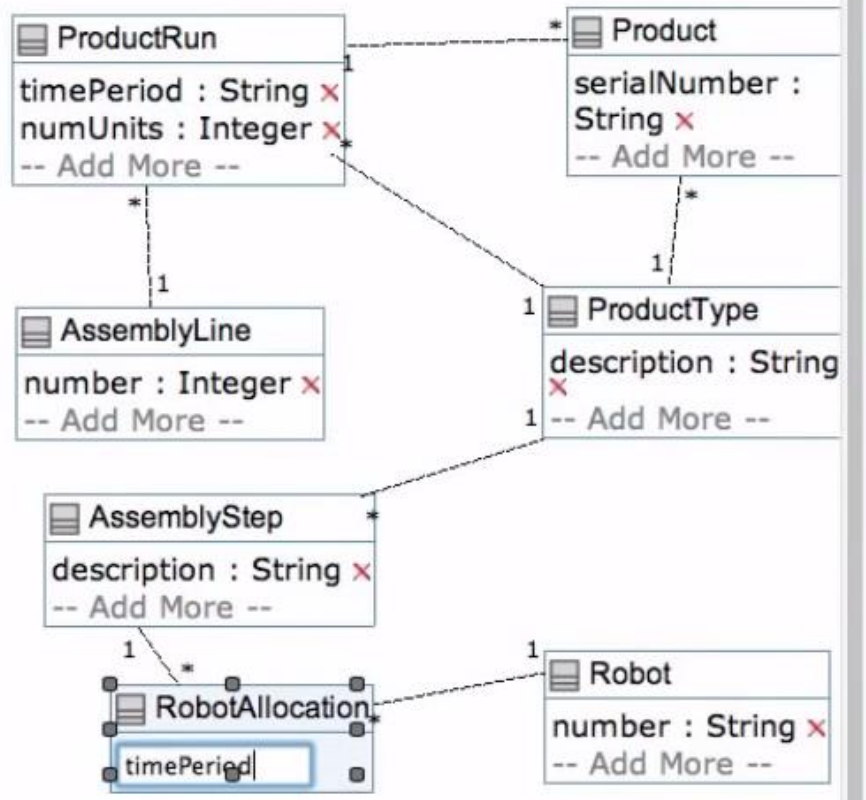
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code

OPTIONS



6:33 / 7:00

Manufacturing Plant Controller 5 - Finishing Off.mp4

Draw on the right, write (Umple) model code on the left. Generate Java, PHP or Ruby code from your models.

Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
8  class ProductType
9  {
10   description;
11 }
12
13 class AssemblyLine
14 {
15   Integer number;
16   1 -- * ProductRun;
17 }
18
19 class ProductRun
20 {
21   timePeriod;
22   Integer numUnits;
23   * -- 1 ProductType;
24 }
25
26 class AssemblyStep
27 {
28   description;
29 }
30
31 class AssemblyStep
32 {
33     * -- 1 ProductType;
34 }
35
36 class Robot
37 {
38 }
39
40 class RobotAllocation
41 {
42   timePeriod;
43   * -- 1 Robot;
```

SAVE & RESET

TOOLS

LOAD

Select Example

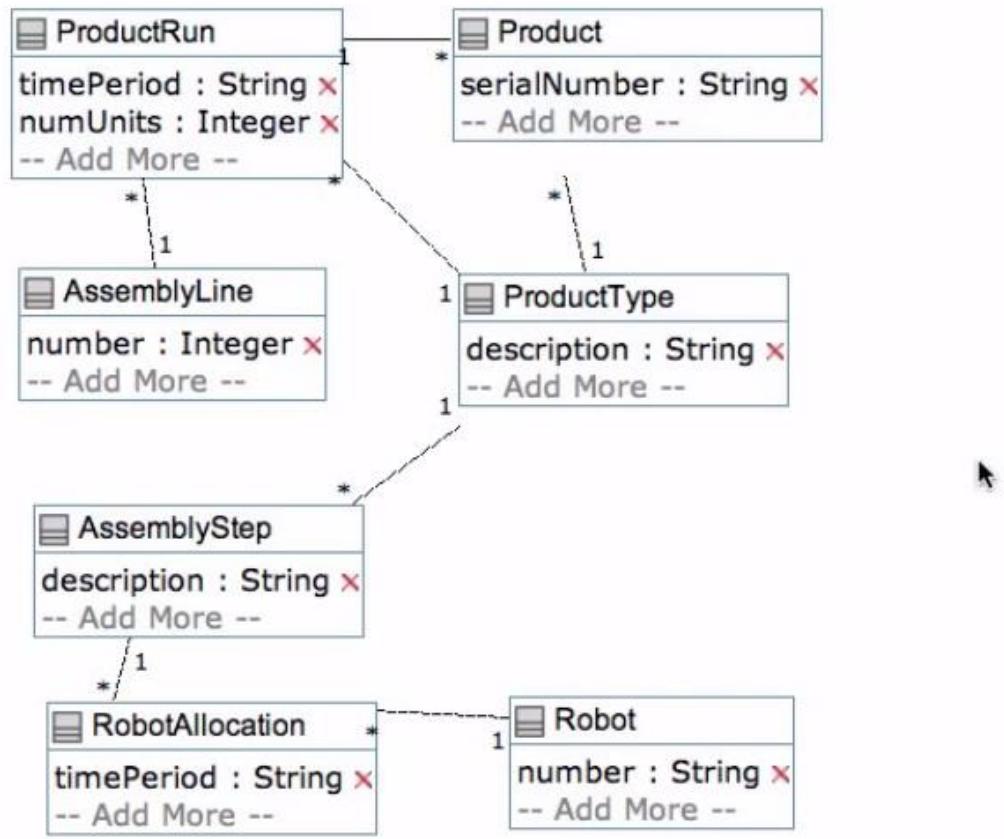
DRAW

Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code



0:29 / 11:53

Cada produto é composto por peças. As peças podem ser compradas de fornecedores, ou podem ser compostas de produtos menores que foram construídos na fábrica (em produtos corridas anteriores). Em cada etapa de fabricação, um dado subconjunto destas partes é montado. Peças esperando para a montagem são mantidos em caixas numeradas; os robôs sabem qual caixas devem acessar para pegar as peças necessárias.

Manufacturing Plant Controller 5 - Finishing Off.mp4

Draw on the right, write (Umple) model code on the left. Generate Java, PHP or Ruby code from your models.

Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
8  class ProductType
9  {
10   description;
11 }
12
13 class AssemblyLine
14 {
15   Integer number;
16   1 -- * ProductRun;
17 }
18
19 class ProductRun
20 {
21   timePeriod;
22   Integer numUnits;
23   * -- 1 ProductType;
24 }
25
26 class AssemblyStep
27 {
28   description;
29 }
30
31 class AssemblyStep
32 {
33     * -- 1 ProductType;
34 }
35
36 class Robot
37 {
38 }
39
40 class RobotAllocation
41 {
42   timePeriod;
43   * -- 1 Robot;
```

SAVE & RESET

TOOLS

LOAD

Select Example

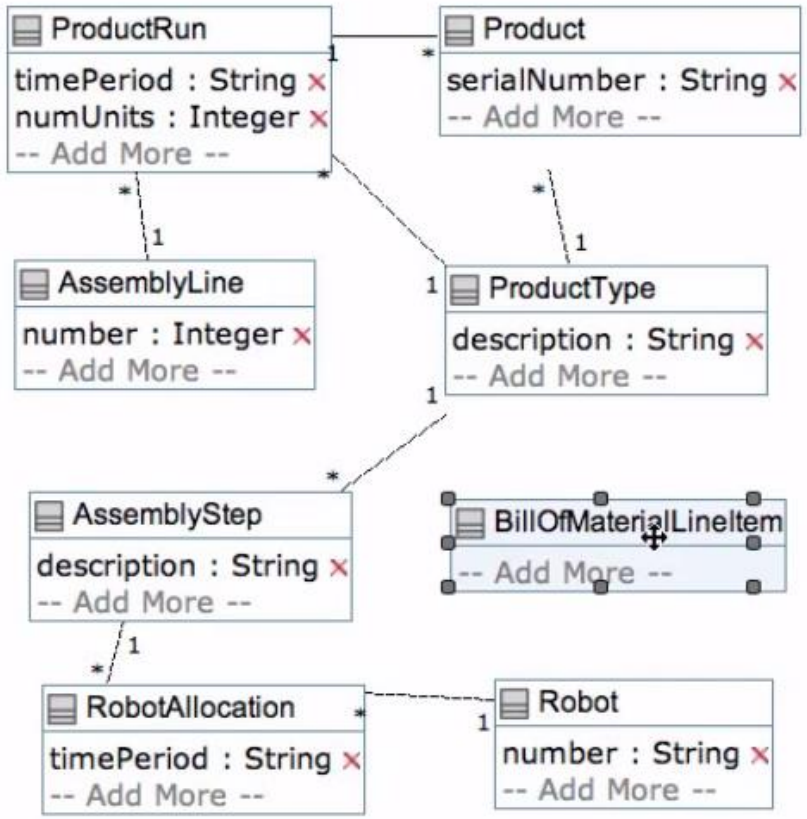
DRAW

Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code



1:57 / 11:53

Manufacturing Plant Controller 5 - Finishing Off.mp4

Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6  }
7
8  class ProductType
9  {
10   description;
11 }
12
13 class AssemblyLine
14 {
15   Integer number;
16   1 -- * ProductRun;
17 }
18
19 class ProductRun
20 {
21   timePeriod;
22   Integer numUnits;
23   * -- 1 ProductType;
24 }
25
26 class AssemblyStep
27 {
28   description;
29 }
30
31 class AssemblyStep
32 {
33     * -- 1 ProductType;
34 }
35
36 class Robot
37 {
38 }
```

SAVE & RESET

TOOLS

LOAD

Select Example

DRAW

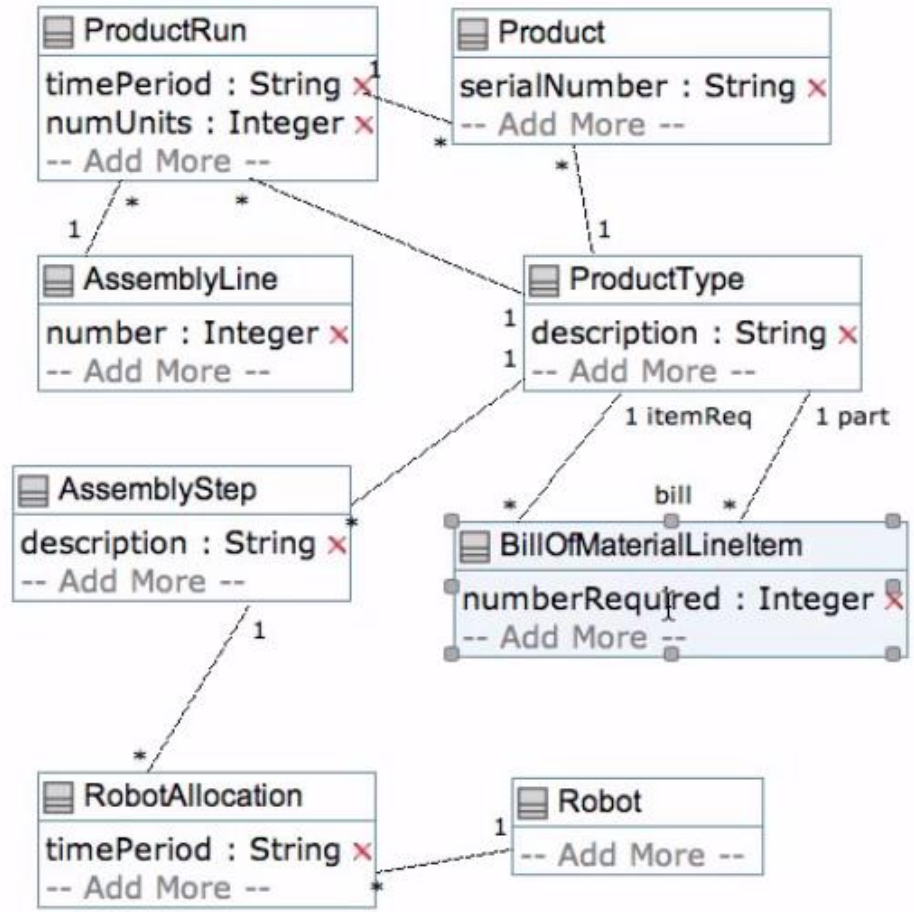
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code

OPTIONS



Manufacturing Plant Controller 5 - Finishing Off.mp4

Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6  }
7
8  class ProductType
9  {
10   description;
11 }
12
13 class AssemblyLine
14 {
15   Integer number;
16   1 -- * ProductRun;
17 }
18
19 class ProductRun
20 {
21   timePeriod;
22   Integer numUnits;
23   * -- 1 ProductType;
24 }
25
26 class AssemblyStep
27 {
28   description;
29 }
30
31 class AssemblyStep
32 {
33     * -- 1 ProductType;
34 }
35
36 class Robot
37 {
38 }
```

SAVE & RESET

TOOLS

LOAD

Select Example

DRAW

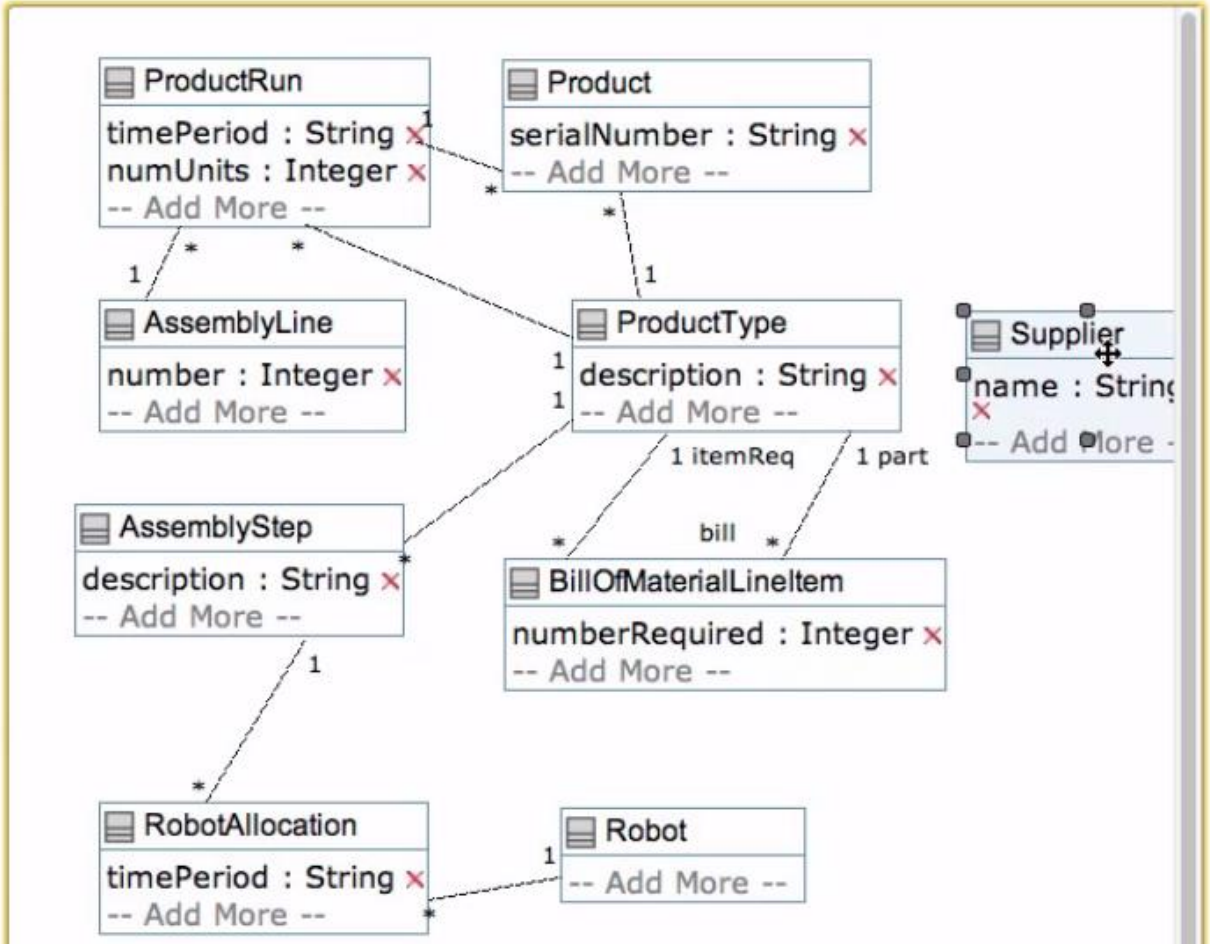
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code

OPTIONS



5:34 / 11:53

Line= 1 Changes at this URL are saved

```
  timePeriod;
  Integer numUnits;
  * -- 1 ProductType;
}

class AssemblyStep
{
  description;
}

class AssemblyStep
{
   * -- 1 ProductType;
}

class Robot
{
}

class RobotAllocation
{
  timePeriod;
  * -- 1 Robot;
  * -- 1 AssemblyStep;
}

class BillOfMaterialLineItem
{
  Integer numberRequired;
  * -- 1 ProductType itemReq;
  * bill -- 1 ProductType part;
}

class Supplier
{
  name;
}
```

SAVE & RESET

TOOLS

LOAD

Select Example

DRAW

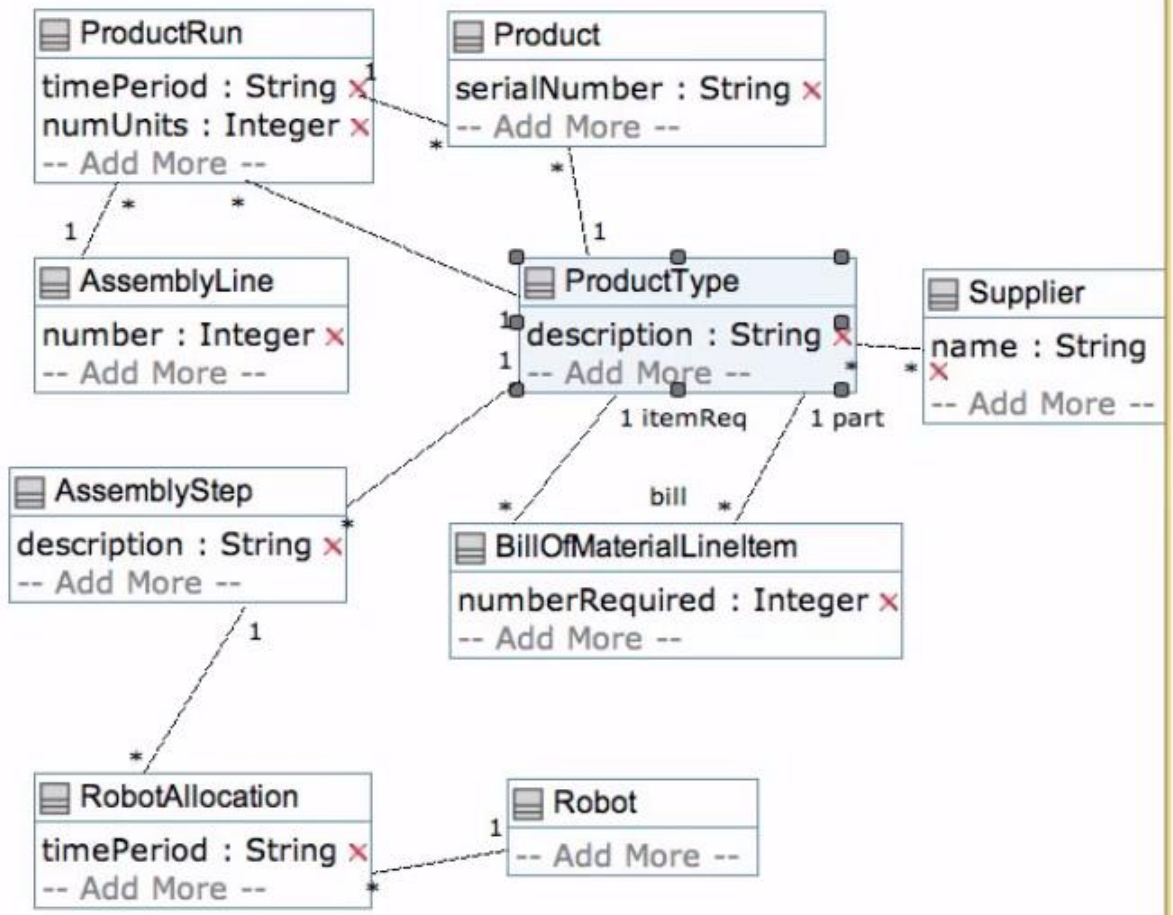
Class
Association
Generalization
Delete
Undo
Redo
Sync Diagram

GENERATE

Java Code

Generate Code

OPTIONS



Manufacturing Plant Controller 5 - Finishing Off.mp4

Draw on the right, write (Umple) model code on the left. Generate Java, PHP or Ruby code from your models.
Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 6 Changes at this URL are saved

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6    * -- 0..1 Supplier;
7  }
8
9  class ProductType
10 {
11   description;
12   * -- * Supplier;
13 }
14
15 class AssemblyLine
16 {
17   Integer number;
18   1 -- * ProductRun;
19 }
20
21 class ProductRun
22 {
23   timePeriod;
24   Integer numUnits;
25   * -- 1 ProductType;
26 }
27
28 class AssemblyStep
29 {
30   description;
31 }
32
33 class AssemblyStep
34 {
35     * -- 1 ProductType;
36 }
37
```

SAVE & RESET

TOOLS

LOAD

Select Example

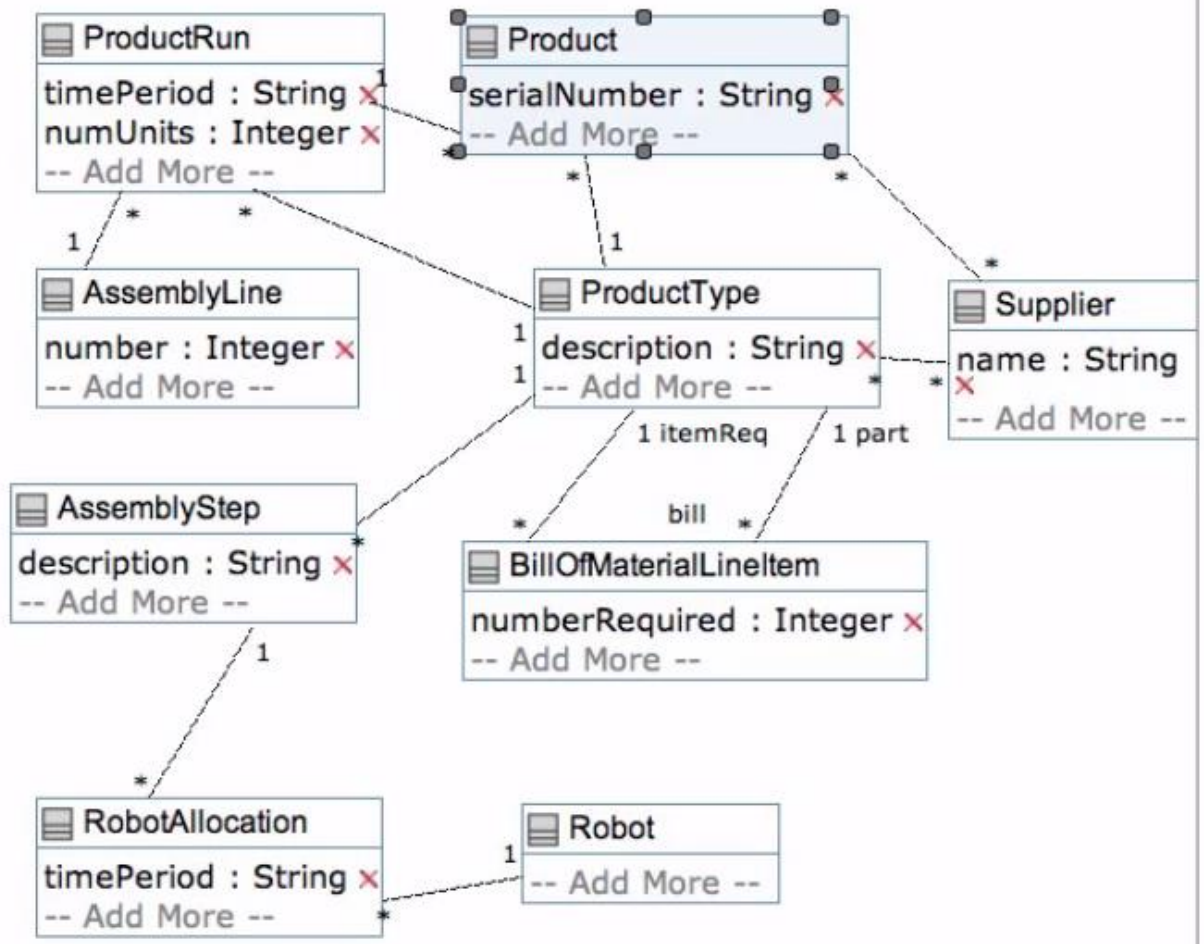
DRAW

- Class
- Association
- Generalization
- Delete
- Undo
- Redo
- Sync Diagram

GENERATE

Java Code

Generate Code



Draw on the right, write (Umple) model code on the left. Generate Java, PHP or Ruby code from your models.
Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 1 Changes at this URL are saved

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6    * -- 0..1 Supplier maker;
7  }
8
9  class ProductType
10 {
11   description;
12   * -- * Supplier;
13 }
14
15 class AssemblyLine
16 {
17   Integer number;
18   1 -- * ProductRun;
19 }
20
21 class ProductRun
22 {
23   timePeriod;
24   Integer numUnits;
25   * -- 1 ProductType;
26 }
27
28 class AssemblyStep
29 {
30   description;
31 }
32
33 class AssemblyStep
34 {
35     * -- 1 ProductType;
36 }
37
```

SAVE & RESET

TOOLS

OPTIONS

SHOW VIEW

- [x] Canvas
- [x] Text Editor
- [ ] Layout Editor

PREFERENCES

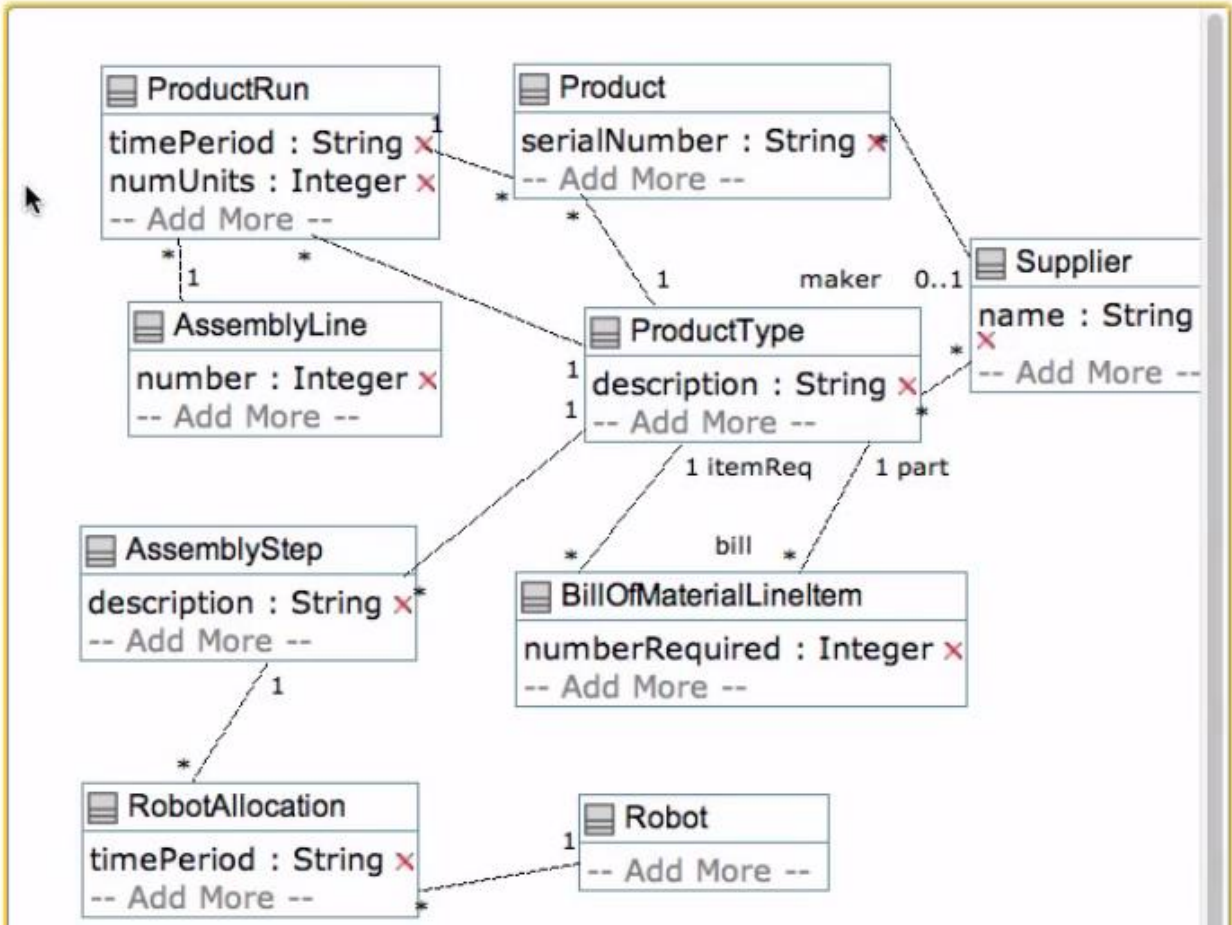
- [ ] Photo Ready
- [ ] Manual Sync

CANVAS SIZE

- Smaller
- Larger

OTHER VIEWS

- Yuml View
- Simulate



Line= 67 Changes at this URL are saved

```
31 }
32
33 class AssemblyStep
34 {
35    * -- 1 ProductType;
36 }
37
38 class Robot
39 {
40 }
41
42 class RobotAllocation
43 {
44   timePeriod;
45   * -- 1 Robot;
46   * -- 1 AssemblyStep;
47 }
48
49 class BillOfMaterialLineItem
50 {
51   Integer numberRequired;
52   * -- 1 ProductType itemReq;
53   * bill -- 1 ProductType part;
54 }
55
56 class Supplier
57 {
58   name;
59 }
60
61 class Order
62 {
63 }
64
65 class Bin
66 {
67
68 }
```

SAVE & RESET

TOOLS

OPTIONS

SHOW VIEW

- [x] Canvas
- [x] Text Editor
- [ ] Layout Editor

PREFERENCES

- [ ] Photo Ready
- [x] Manual Sync

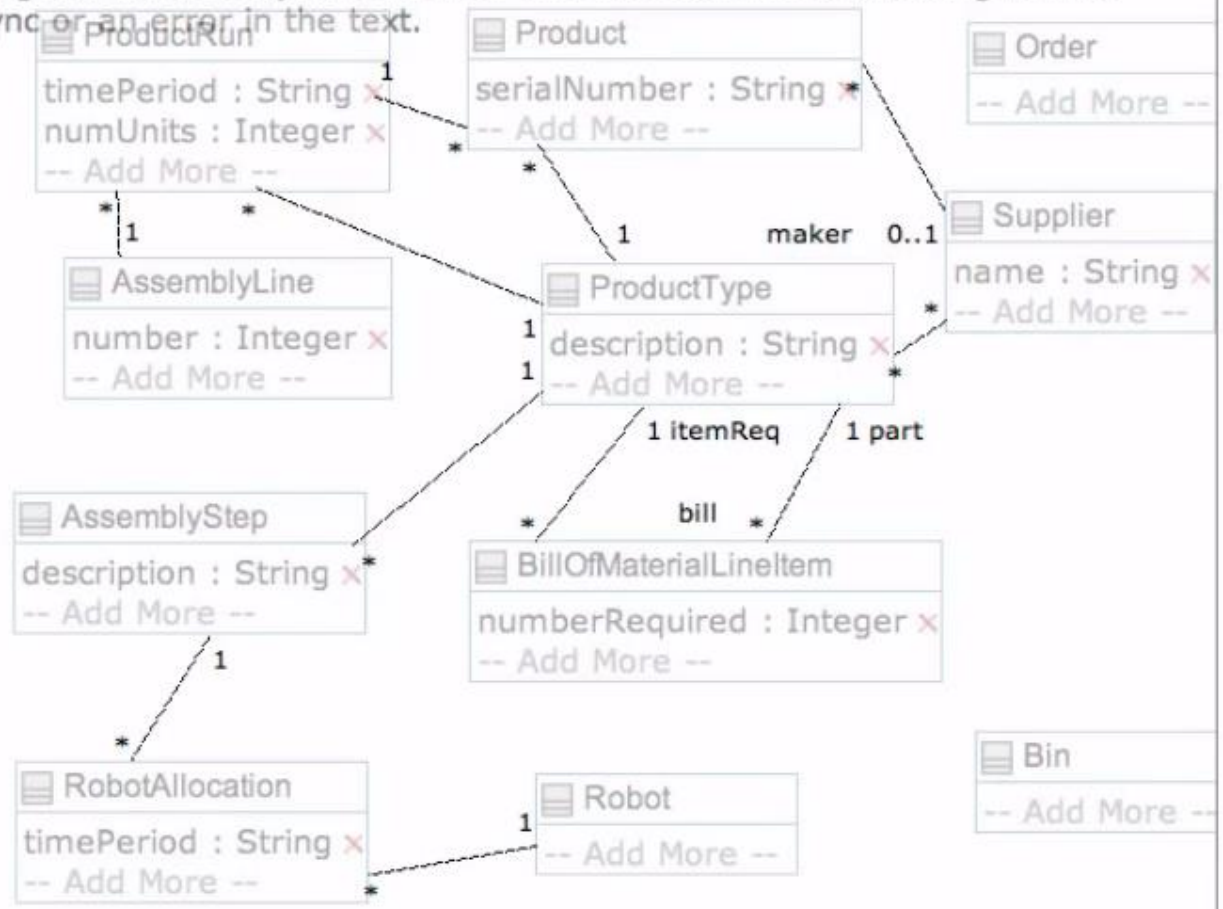
CANVAS SIZE

- Smaller
- Larger

OTHER VIEWS

- Yuml View
- Simulate

Diagram is out of synchronization with the text due to selecting Manual Sync or an error in the text.



Manufacturing Plant Controller 5 - Finishing Off.mp4

Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 13 Changes at this URL are saved

```
  * -- 1 ProductType;
  * -- 1 ProductRun;
  * -- 0..1 Supplier maker;
}

class ProductType
{
  description;
  * -- * Supplier;
  0..1 -- * Bin;
}

class AssemblyLine
{
  Integer number;
  1 -- * ProductRun;
}

class ProductRun
{
  timePeriod;
  Integer numUnits;
  * -- 1 ProductType;
}

class AssemblyStep
{
  description;
}

class AssemblyStep
{
   * -- 1 ProductType;
}

class Robot
{
}
```

SAVE & RESET

TOOLS

OPTIONS

SHOW VIEW

- [x] Canvas
- [x] Text Editor
- [ ] Layout Editor

PREFERENCES

- [ ] Photo Ready
- [x] Manual Sync

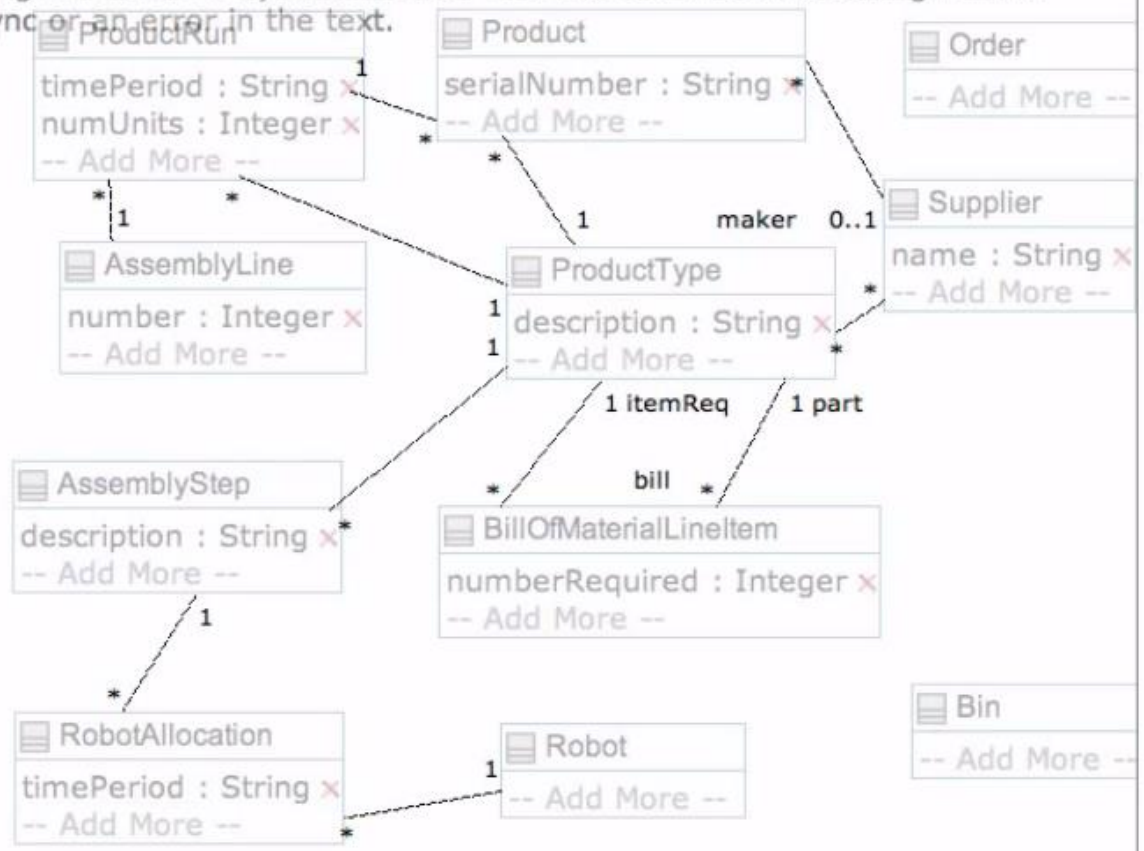
CANVAS SIZE

- Smaller
- Larger

OTHER VIEWS

- Yuml View
- Simulate

Diagram is out of synchronization with the text due to selecting Manual Sync or an error in the text.



9:40 / 11:53

Manufacturing Plant Controller 5 - Finishing Off.mp4 Visit the User Manual or the Umple Home Page for help. Download Umple Report an Issue

Line= 64 Changes at this URL are saved

```
31  description;
32 }
33
34 class AssemblyStep
35 {
36    * -- 1 ProductType;
37 }
38
39 class Robot
40 {
41 }
42
43 class RobotAllocation
44 {
45   timePeriod;
46  * -- 1 Robot;
47  * -- 1 AssemblyStep;
48 }
49
50 class BillOfMaterialLineItem
51 {
52  Integer numberRequired;
53  * -- 1 ProductType itemReq;
54  * bill -- 1 ProductType part;
55 }
56
57 class Supplier
58 {
59  name;
60 }
61
62 class Order
63 {
64  0..1 -- * Product;
65 }
66
67 class Bin
68 {
```

SAVE & RESET

TOOLS

OPTIONS

SHOW VIEW

- [x] Canvas
- [x] Text Editor
- [ ] Layout Editor

PREFERENCES

- [ ] Photo Ready
- [x] Manual Sync

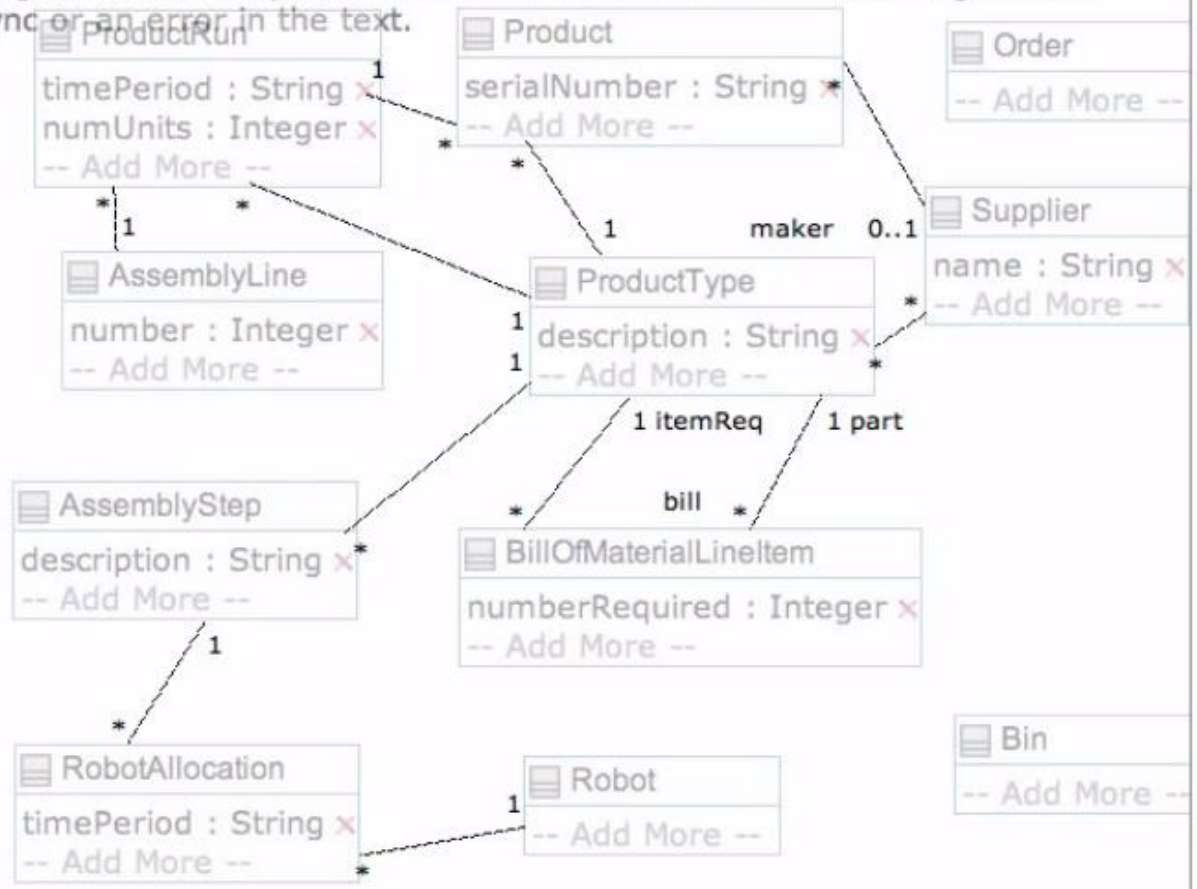
CANVAS SIZE

- Smaller
- Larger

OTHER VIEWS

- Yuml View
- Simulate

Diagram is out of synchronization with the text due to selecting Manual Sync or an error in the text.



number

Line= 1 Changes at this URL are saved

```
1  class Product
2  {
3    serialNumber;
4    * -- 1 ProductType;
5    * -- 1 ProductRun;
6    * -- 0..1 Supplier maker;
7  }
8
9  class ProductType
10 {
11   description;
12   * -- * Supplier;
13   0..1 -- * Bin;
14 }
15
16 class AssemblyLine
17 {
18   Integer number;
19   1 -- * ProductRun;
20 }
21
22 class ProductRun
23 {
24   timePeriod;
25   Integer numUnits;
26   * -- 1 ProductType;
27 }
28
29 class AssemblyStep
30 {
31   description;
32 }
33
34 class AssemblyStep
35 {
36     * -- 1 ProductType;
37 }
38
```

SAVE & RESET

TOOLS

OPTIONS

SHOW VIEW

- [x] Canvas
- [x] Text Editor
- [ ] Layout Editor

PREFERENCES

- [ ] Photo Ready
- [ ] Manual Sync

CANVAS SIZE

- Smaller
- Larger

OTHER VIEWS

- Yuml View
- Simulate