



Material Didático sobre Robocode

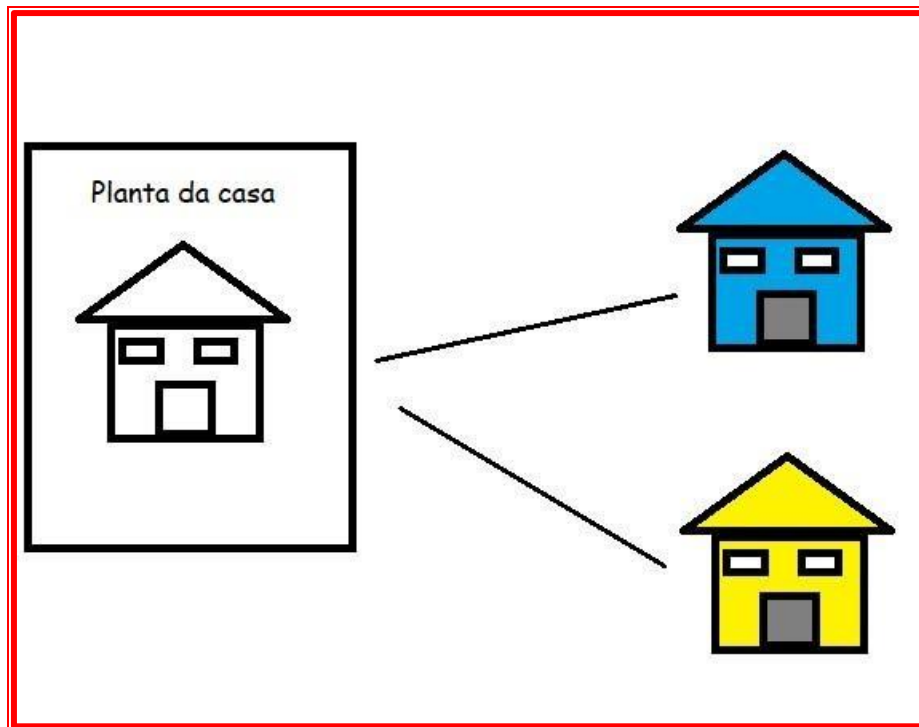
1 Conceitos básicos sobre as técnicas de Orientação a Objetos

Para que possamos utilizar o Robocode para construção do nosso Robô, usaremos a linguagem de programação Java. E Java é uma linguagem de programação orientada a objetos. Para facilitar o entendimento daqueles que ainda não possuem familiaridade com essa linguagem de programação, vamos trazer alguns conceitos sobre orientação a objetos neste e em novos materiais que estarão disponíveis no site Robótica Paula Souza. Quando abrimos um novo projeto para criar nosso robô, estamos criando na verdade uma classe. Nesta classe definimos especificações para podermos criar instâncias, ou seja, um espaço para podermos definir características e ações, que são nossos objetos.

As classes possuem uma série de atributos e comportamentos em comum, mas, não iguais, podendo variar nos valores armazenados desses atributos e em como realizam seus comportamentos, ações.

Um exemplo clássico para entendermos **classes**, **instância** e **objeto** é o da planta de uma casa. A planta é o nosso projeto em que definimos os **atributos** (valores armazenados) e **métodos** (comportamento) no papel, mas para que a casa seja habitável e possua de fato atributos e métodos, temos que construir instâncias a partir dessa planta, para podermos morar nela. A partir de uma mesma planta (classe) podemos construir instâncias diferentes, com atributos diferentes, como por exemplo, a cor da casa. Posso ter um objeto casa instanciado com a cor azul, ou amarelo, que foram construídos a partir da mesma classe (planta).

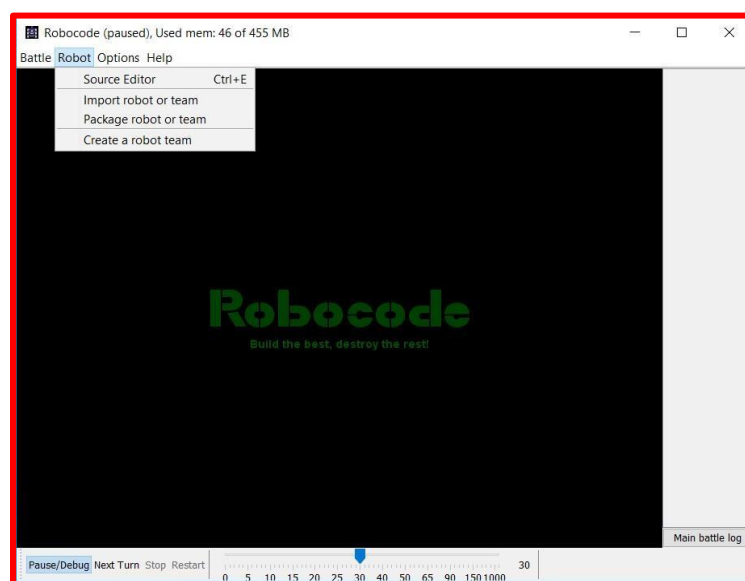
Administração Central
CETEC Capacitações



Portanto, classe e objeto não são sinônimos, pois os **objetos** são criados a partir das definições de uma classe.

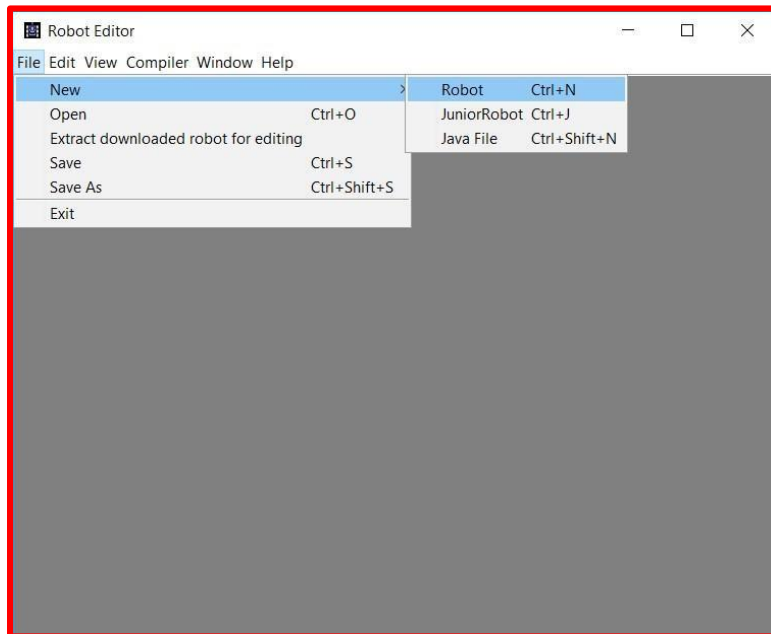
Vamos criar um novo robô chamado **AprendaClasses** e colocá-lo também na pasta competição (**package**).

Para isso, abra o **Robocode** e selecione a opção **Source Editor** do menu **Robot**

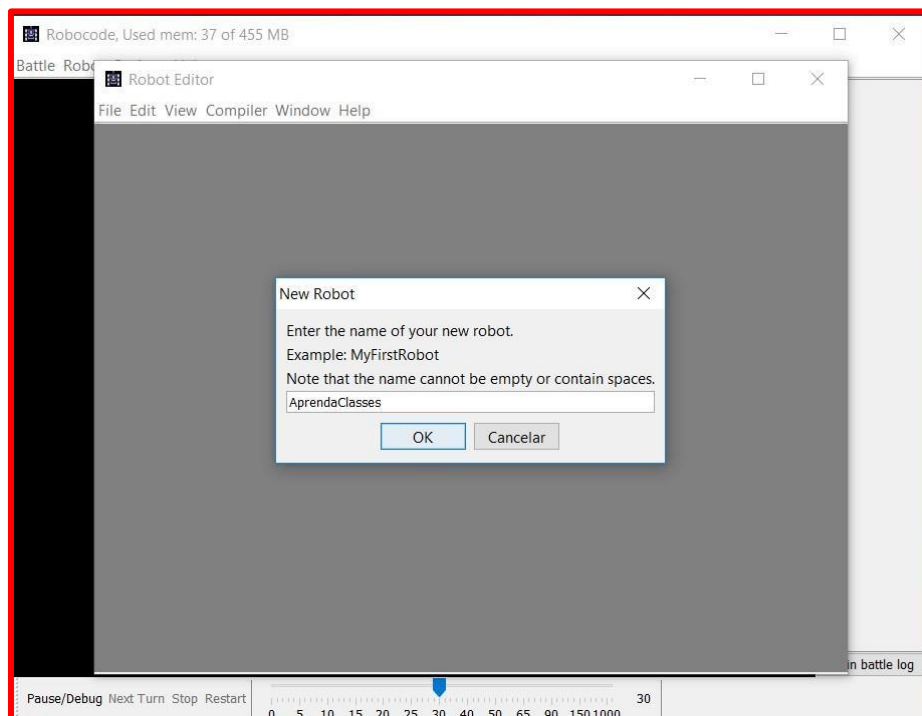


Administração Central CETEC Capacitações

No editor, selecione a opção **New** do Menu **File** e em seguida selecione a opção **Robot**.



Vamos nomear nosso robô **AprendaClasses**



Administração Central
CETEC Capacitações

E por último, vamos colocá-lo no pacote **competicao**

Enter a short package name for your new robot and without spaces (lower-case letters are preferred). Your initials will work well here.

Your robot will be put into this package to avoid name conflict with other robots.

The package name is used to identify your robot(s) in the game, especially if you want to let your robot(s) participate in competitions like e.g. RoboRumble@Home.

Hence, you should enter the same package name for all of your robots.

Note that the package name cannot be empty or contain spaces.

competicao

OK Cancelar

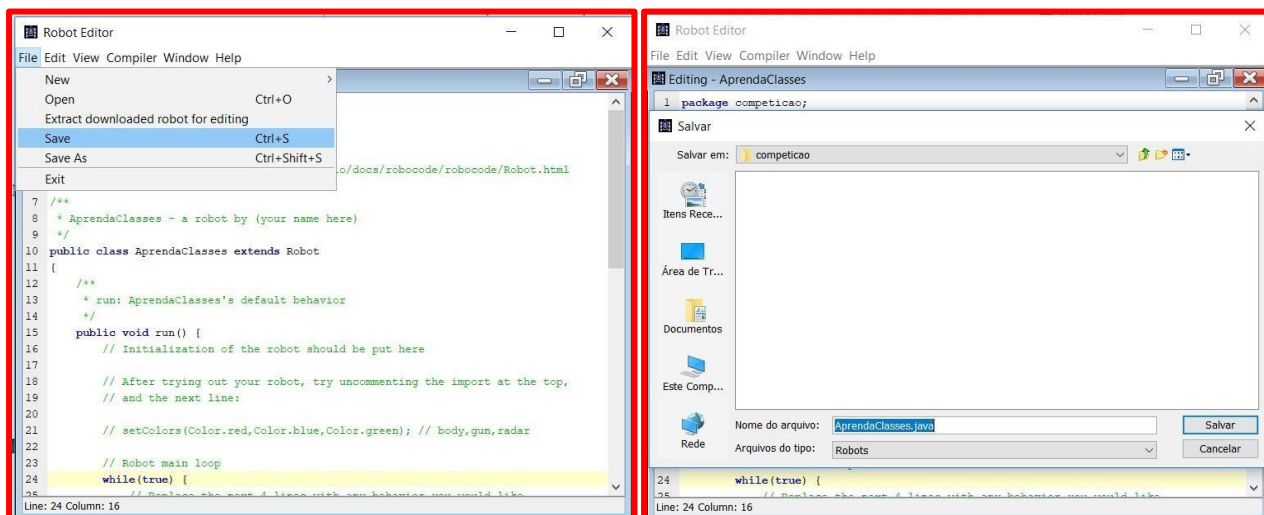
Observe que o código fonte já possui outras **classes** importadas ao seu projeto, incluindo a **herança** com a classe **Robot**, métodos, eventos, já definidos como está especificado no material Conhecendo o Robô:

http://www.robotica.cpsctec.com.br/material/03_Conhecendo_o_Robo.pdf

```
Robot Editor
File Edit View Compiler Window Help
Editing - AprendaClasses
1 package competicao;
2 import robocode.*;
3 //import java.awt.Color;
4
5 // API help : https://robocode.sourceforge.io/docs/robocode/robocode/Robot.html
6
7 /**
8  * AprendaClasses - a robot by (your name here)
9  */
10 public class AprendaClasses extends Robot
11 {
12     /**
13      * run: AprendaClasses's default behavior
14      */
15     public void run() {
16         // Initialization of the robot should be put here
17
18         // After trying out your robot, try uncommenting the import at the top,
19         // and the next line:
20
21         // setColors(Color.red,Color.blue,Color.green); // body,gun,radar
22
23         // Robot main loop
24         while(true) {
25             // Replace the next 4 lines with any behavior you would like
```

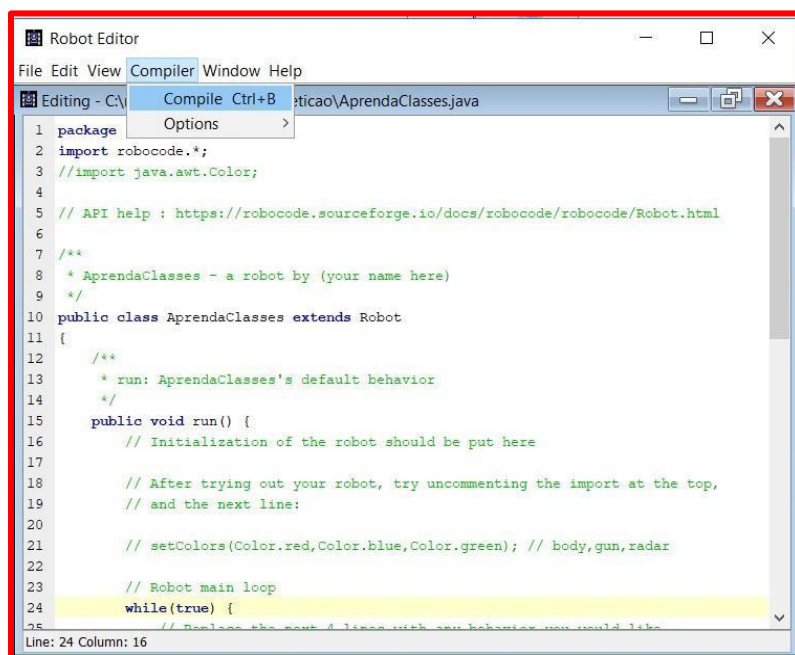
Administração Central CETEC Capacitações

Vamos salvar nosso Robô AprendaClasses



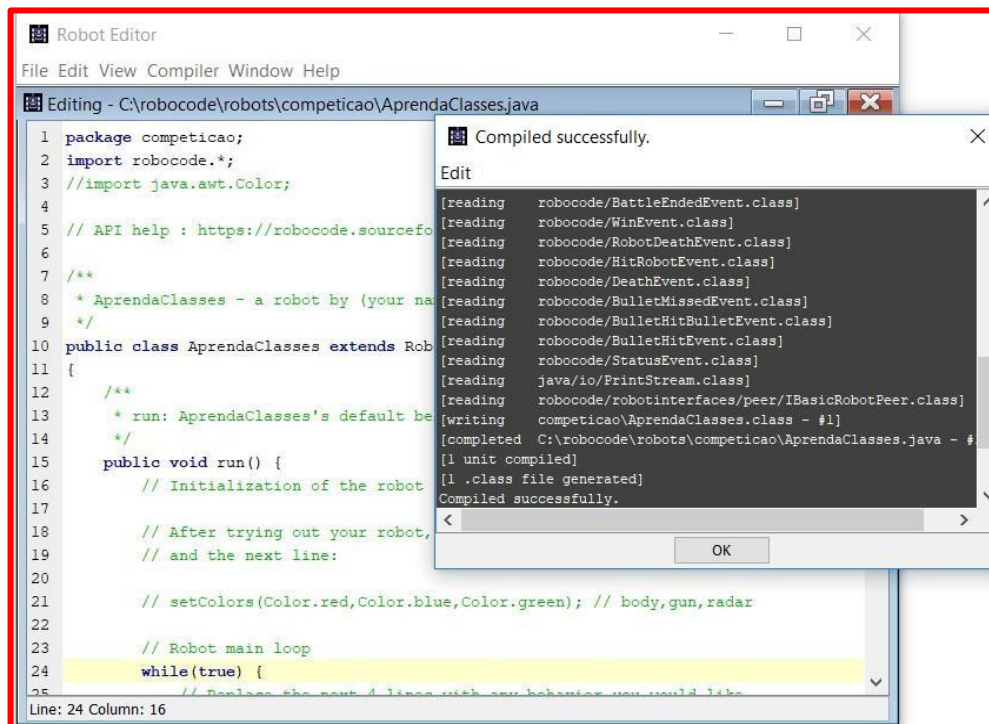
Observe que ao criá-lo já tínhamos definido o **pacote**, ele já é direcionado para a pasta **competicao** e o seu **tipo** definido como **Robots**, que é o local onde estão todos os Robôs do Robocode. Ele irá salvar nosso arquivo com extensão **.java**.

Agora vamos compilar nosso robô **AprendaClasses** para que possamos utilizá-lo futuramente e com isso será gerado o arquivo com extensão **.class**.



Administração Central CETEC Capacitações

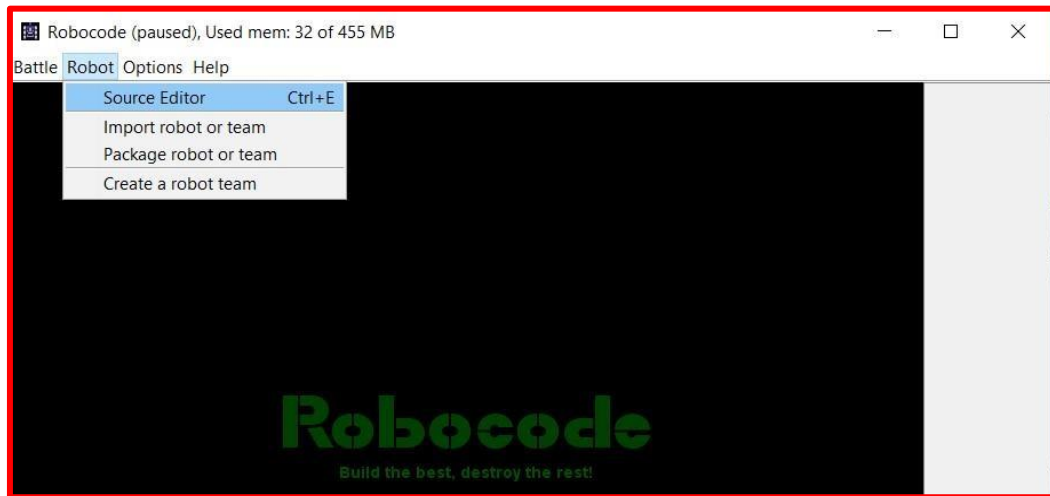
Se tudo estiver correto, teremos a tela a seguir: **Compilado com sucesso.**



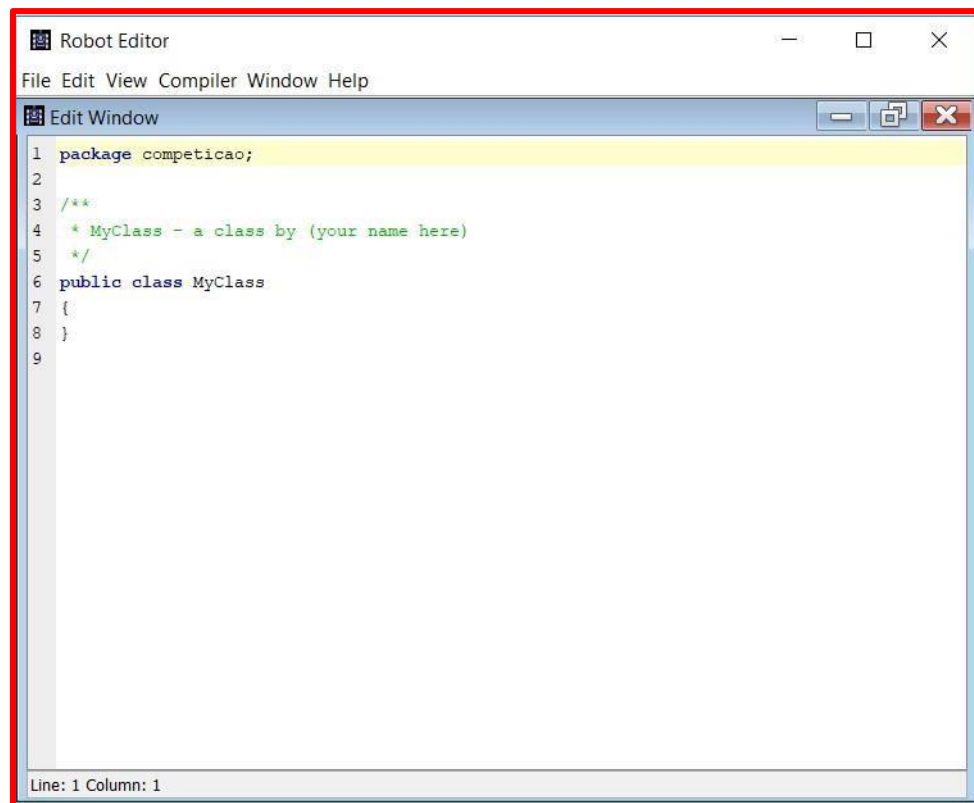
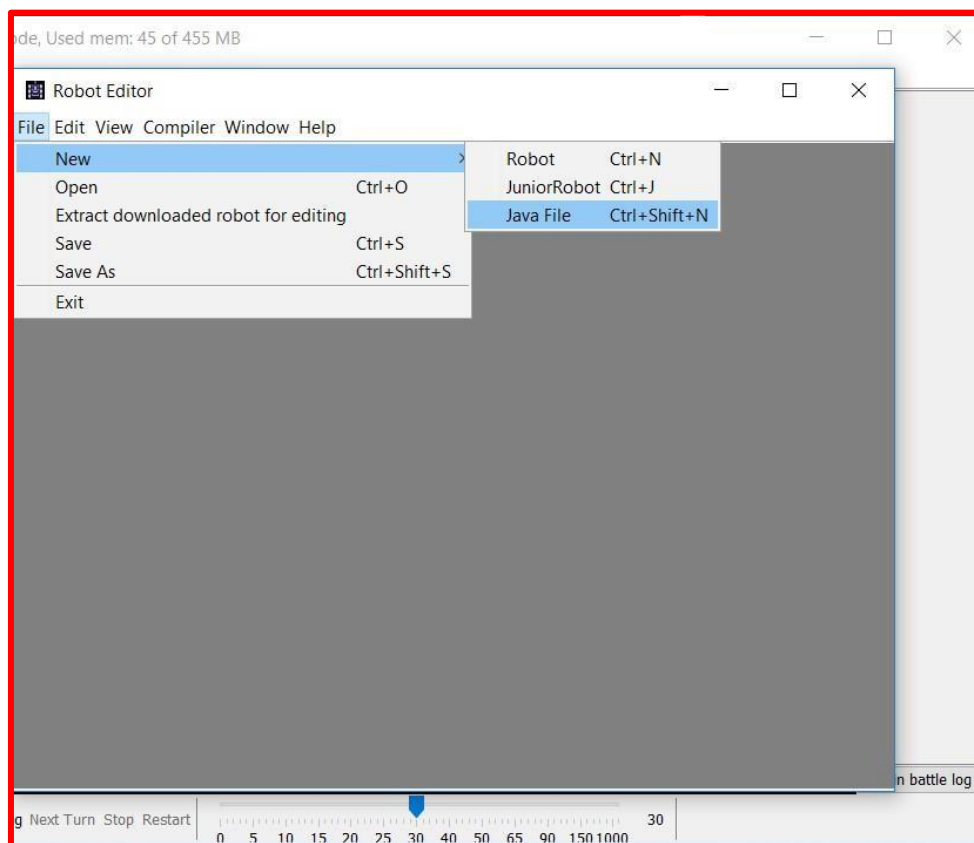
2 Criando um Robô do zero

Agora vamos abrir um novo arquivo fonte no editor (**Source Editor**), mas dessa vez vamos abrir um arquivo Java (escolher a opção **Java File**) para criarmos nosso robô do zero e entender os conceitos de Orientação a objetos.

Administração Central CETEC Capacitações



Administração Central CETEC Capacitações

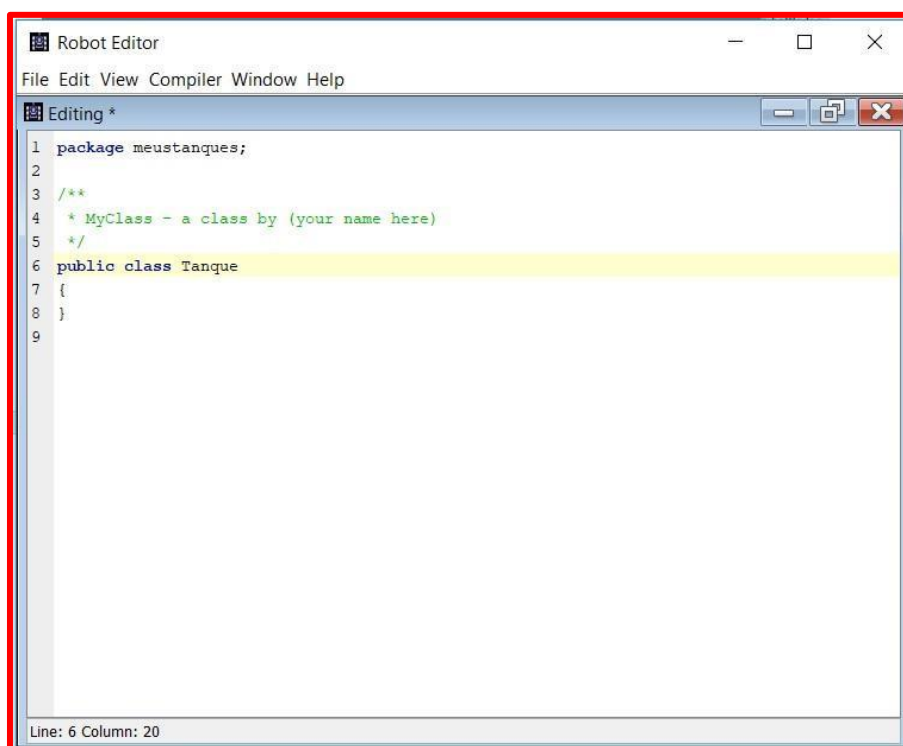


Administração Central CETEC Capacitações

Observe que como não havíamos definido nome da classe e nem pacote, ele trouxe o pacote aberto anteriormente e o nome da **classe padrão MyClass**.

Na classe criada, vamos definir alguns atributos. Ou seja, características de algumas partes do nosso **Tanque**, que também será o nome da nossa classe. Vamos também definir um nome para o nosso pacote, ou seja, local onde colocaremos todas as classes que quisermos criar, e para que outras pessoas possam utilizá-las, deverá utilizar esse pacote, que é uma espécie de biblioteca onde temos várias classes definidas com seus atributos e métodos. Chamaremos nosso pacote de **meustanques**.

Para isso, vamos alterar nosso código fonte:



```
Robot Editor
File Edit View Compiler Window Help
Editing *
1 package meustanques;
2
3 /**
4  * MyClass - a class by (your name here)
5  */
6 public class Tanque
7 {
8 }
9
Line: 6 Column: 20
```

Tudo que iremos programar em nosso tanque, precisa estar dentro do método principal do Robocode que é o **método run()**. É o método que define a inicialização do nosso robô e a configuração que iremos utilizar.

public void run(){

}

Administração Central CETEC Capacitações

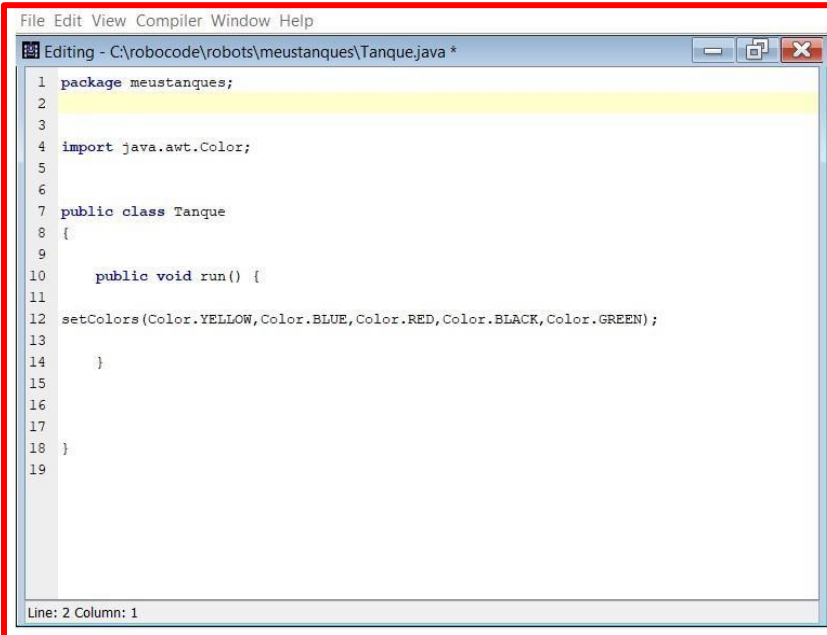
O método é **público** para que as classes tenham acesso a ele, e seu tipo é **void (vazio)**, pois não tem valor de retorno.

Até aqui só definimos o pacote onde encontraremos nosso **Tanque**, o nome da sua classe e o método que faz com que o robô inicialize. Agora dentro da nossa classe vamos definir algumas **características** para o **Tanque**. Se quisermos definir a cor do tanque podemos utilizar o **método setColors()**. Esse método está dentro do pacote **java.awt.Color**, por isso, para usá-lo devemos importá-lo para o nosso robô Tanque: **import java.awt.Color;** O método **setColors()** define 5 parâmetros de cor para o robô, na seguinte ordem: cor do corpo, da arma, do radar, da bala e do arco de varredura ao mesmo tempo.

Exemplo: **setColors(Color.YELLOW , Color.BLUE , Color.RED , Color.BLACK , Color.GREEN);**

Como parâmetro, usamos a classe **Color** seguido do ponto e do valor que é a cor que deseja definir para a parte do tanque correspondente.

Até agora nosso código fonte estará assim:

A screenshot of a Java IDE window titled "Editing - C:\robocode\robots\meustanques\Tanque.java *". The code is as follows:

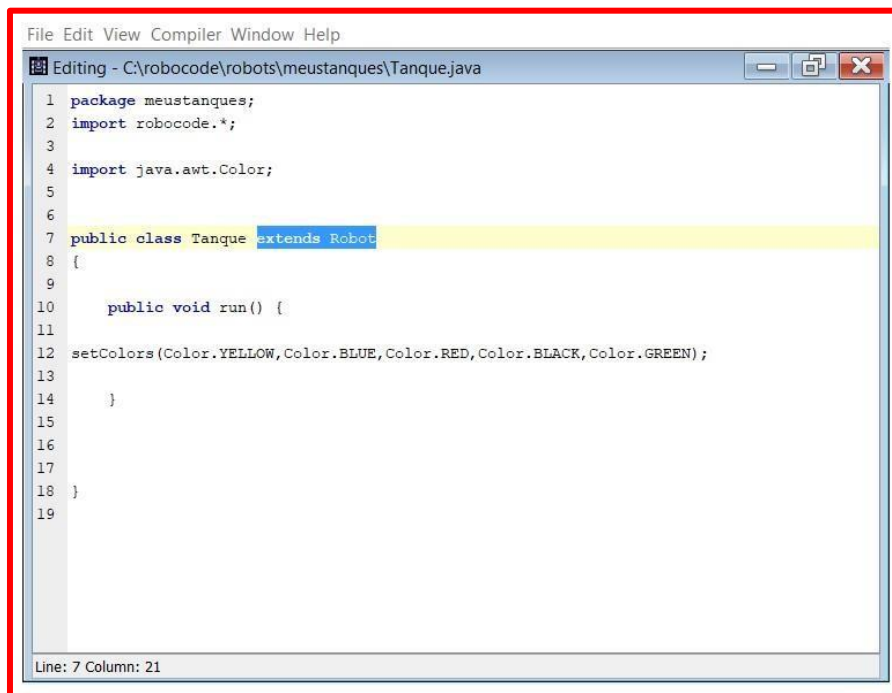
```
1 package meustanques;
2
3
4 import java.awt.Color;
5
6
7 public class Tanque
8 {
9
10     public void run() {
11
12         setColors(Color.YELLOW, Color.BLUE, Color.RED, Color.BLACK, Color.GREEN);
13     }
14 }
15
16
17
18 }
19
```

The status bar at the bottom indicates "Line: 2 Column: 1".

Porém, se salvarmos e compilarmos nosso código, ele apresentará um erro.

3 Importando pacotes

Todos os robôs criados no Robocode precisam utilizar algumas classes específicas do pacote **robocode**. Para isso, iremos importar tudo desse pacote: **import robocode.*;** Isso deve ser feito para que possamos utilizar em nossa classe **Tanque** todos os recursos da classe **Robot**. E é aqui que entra o conceito de **herança**. Nossa classe **Tanque** irá herdar características da classe **Robot**, então precisamos usar a palavra reservada **extends** e o nome da superclasse **Robot**. O método **run()** pertence a classe **Robot**, por isso, sempre iremos estender nossas classes a ela, se formos usar características dessa classe. Caso contrário, não poderemos inicializar nosso robô.



```
File Edit View Compiler Window Help
Editing - C:\robocode\robots\meustanques\Tanque.java
1 package meustanques;
2 import robocode.*;
3
4 import java.awt.Color;
5
6
7 public class Tanque extends Robot
8 {
9
10     public void run() {
11
12         setColors(Color.YELLOW, Color.BLUE, Color.RED, Color.BLACK, Color.GREEN);
13
14     }
15
16
17
18 }
19
Line: 7 Column: 21
```

Quando criamos novas classes a partir de outras previamente criadas, chamamos de **herança na POO** (Programação Orientada a Objetos). Essas novas classes são chamadas de subclasses, ou classes derivadas. As classes já existentes, que deram origem às subclasses, são chamadas de superclasses, ou classes base. **Uma subclasse herda métodos e atributos de sua superclasse.**

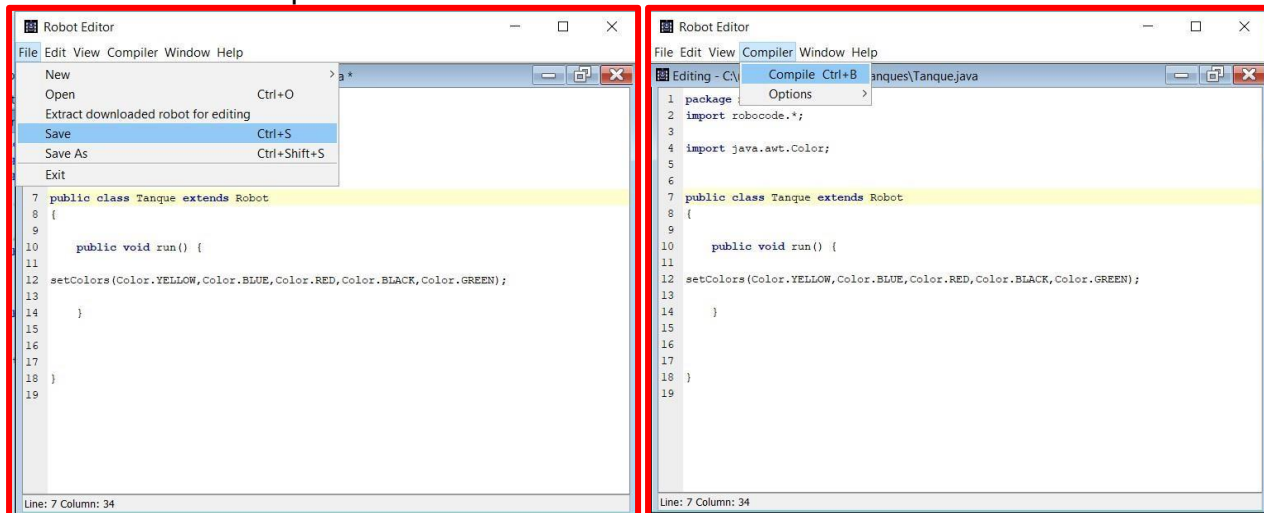
Mesmo utilizando a herança, podemos escrever novamente os métodos e atributos de uma forma mais específica para representar o comportamento do método herdado. No nosso

Administração Central CETEC Capacitações

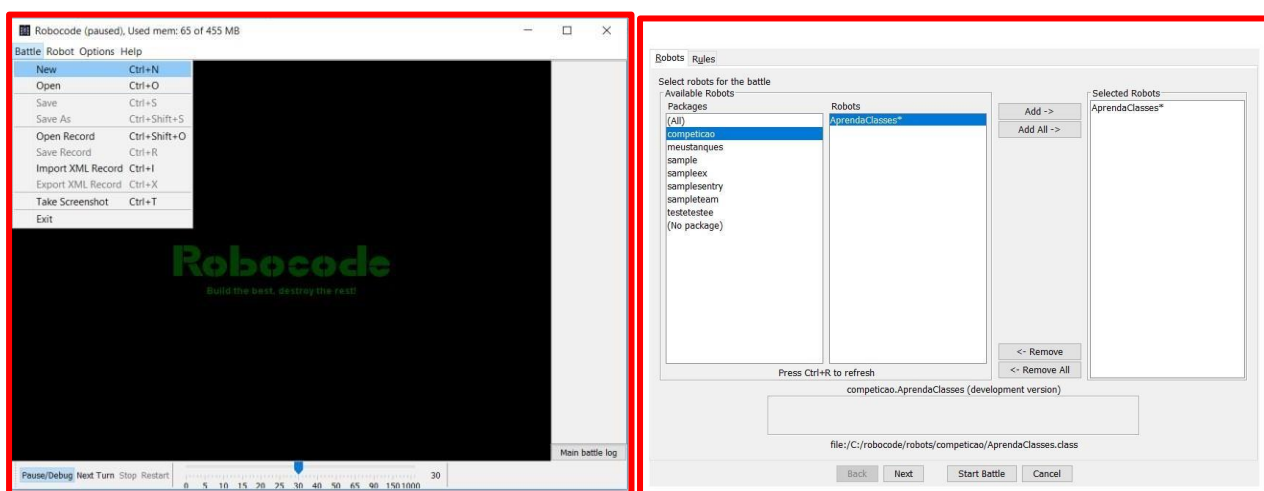
caso estamos usando o método `run()` herdado de `Robot`, que é uma característica comum entre os robôs, mas iremos modificá-lo para atender nossa estratégia de jogo e características específicas que desejamos para o nosso robô.

Até aqui nossa classe possui algumas características específicas definindo as cores das partes do nosso robô. Porém, não tem ações.

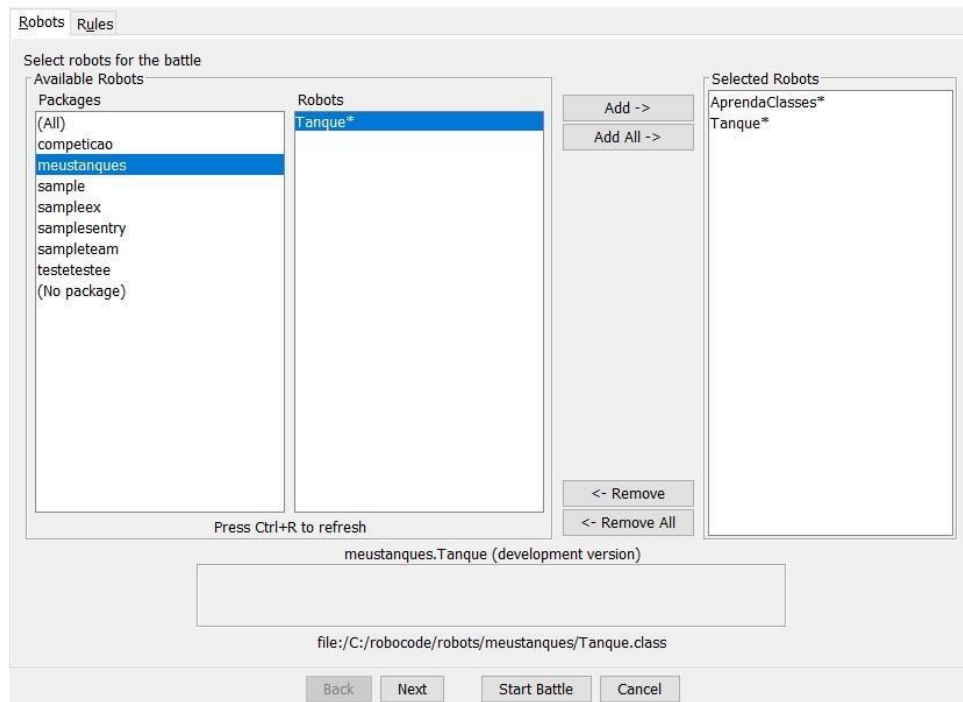
Vamos salvar e compilar nosso robô.



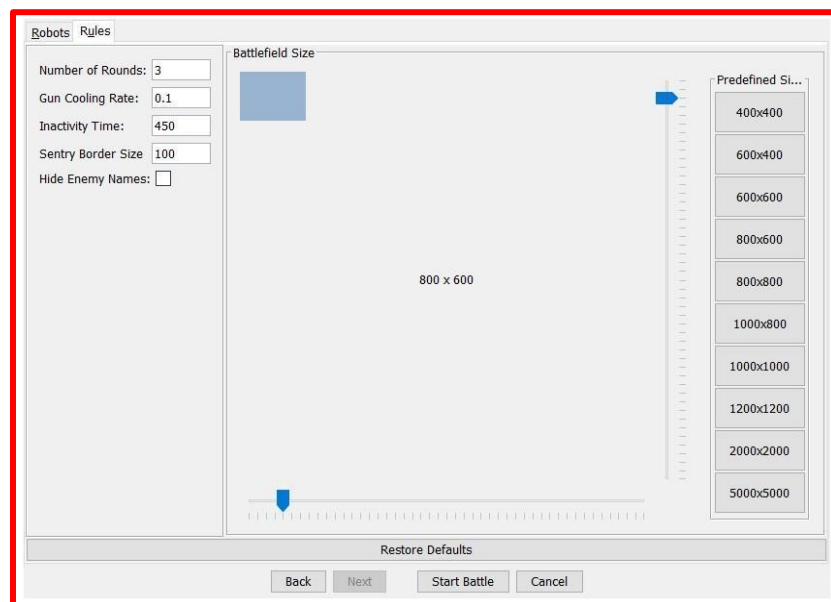
Agora vamos colocá-lo na arena com o robô `AprendaClasses` apresentado no início do material.



Administração Central CETEC Capacitações

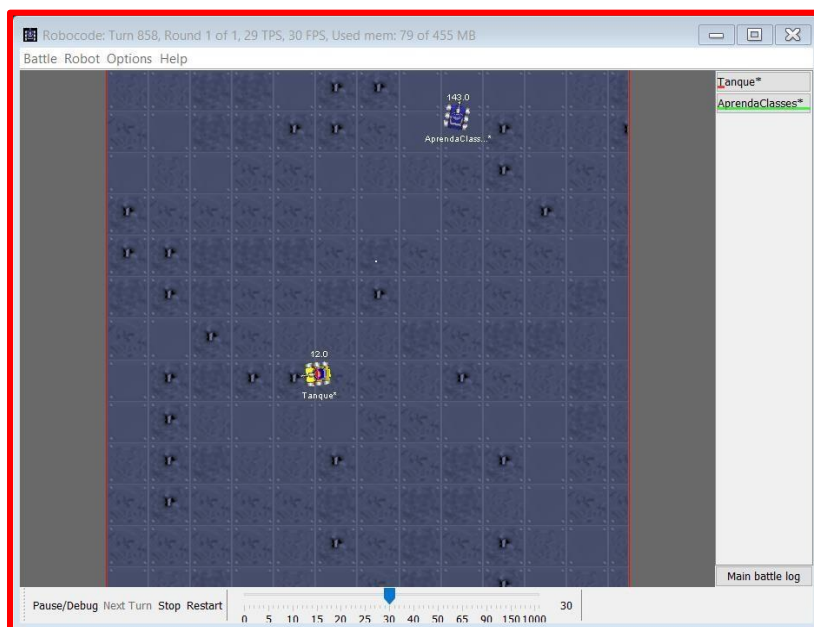


Depois de adicionar os dois robôs, clique em **Next** para configurarmos a arena conforme as regras da competição do Torneio de Robótica Virtual - Robocode.



E pressionar **Start Battle** para iniciarmos a batalha.

Administração Central CETEC Capacitações



Nosso robô não tem movimentos, mas está com uma aparência diferente dos demais robôs! No próximo material traremos mais conceitos de orientação a objetos melhorando a movimentação do nosso robô.

4 Referências

JAVADOCEXAMPLES. **Demos and Usage of java.awt.Graphics.setColor(Color c).** Disponível em: <[http://www.javadoceexamples.com/java/awt/Graphics/setColor\(Color%20c\).html](http://www.javadoceexamples.com/java/awt/Graphics/setColor(Color%20c).html)>. Acesso em 25/03/2019.

ROBOWIKI. **Robocode/My First Robot**. Disponível em: <http://robowiki.net/w/index.php?title=Robocode/My_First_Robot#My_First_Robot>. Acesso em 10/04/2021.

DEVEMEDIA. **Entendendo e aplicando Herança em Java.** Disponível em: <<https://www.devmedia.com.br/entendendo-e-aplicando-heranca-em-java/24544>>. Acesso em 10/04/2021.