WPILib provides separate Robot Drive classes for the most common drive train configurations (differential, mecanum, and Killough). The DifferentialDrive class handles the differential drivetrain configuration. These drive bases typically have two or more in-line traction or omni wheels per side (e.g., 6WD or 8WD) and may also be known as "skid-steer", "tank drive", or "West Coast Drive". The Kit of Parts drivetrain is an example of a differential drive. There are methods to control the drive with 3 different styles ("Tank", "Arcade", or "Curvature"), explained in the article below.

Conventions and Defaults

For more information about conventions and defaults of the DifferentialDrive class see WPILib Drive classes: Conventions and Defaults

Creating a Differential Drive object

C++

```cpp
class Robot
{
    public:
        frc::Spark m_left{1};
        frc::Spark m_right{2};
        frc::DifferentialDrive m_drive{m_left, m_right};
}
```

Java

```java
public class Robot
{
    Spark m_left = new Spark(1);
    Spark m_right = new Spark(2);
    DifferentialDrive m_drive = new DifferentialDrive(m_left, m_right);
}    ```
Multi-Motor Drives

Many FRC drivetrains have more than 1 motor on each side. In order to use these with DifferentialDrive, the motors on each side have to be collected into a single SpeedController, using the SpeedControllerGroup class. The examples below show a 4 motor (2 per side) drivetrain. To extend to more motors, simply create the additional controllers and pass them all into the SpeedController group constructor (it takes an arbitrary number of inputs).

**C++**

```cpp
class Robot
{
  public:
    frc::Spark m_frontLeft{1};
    frc::Spark m_rearLeft{2};
    frc::SpeedControllerGroup m_left{m_frontLeft, m_rearLeft};

    frc::Spark m_frontRight{3};
    frc::Spark m_rearRight{4};
    frc::SpeedControllerGroup m_right{m_frontRight, m_rearRight};

    frc::DifferentialDrive m_drive{m_left, m_right};
}
```

**Java**

```java
public class Robot
{
  Spark m_frontLeft = new Spark(1);
  Spark m_rearLeft = new Spark(2);
  SpeedControllerGroup m_left = new SpeedControllerGroup(m_frontLeft, m_rearLeft);

  Spark m_frontRight = new Spark(3);
  Spark m_rearRight = new Spark(4);
  SpeedControllerGroup m_right = new SpeedControllerGroup(m_frontRight, m_rearRight);
  DifferentialDrive m_drive = new DifferentialDrive(m_left, m_right);
}
```
Drive Modes

The DifferentialDrive class contains 3 drive modes:

- Tank Drive - This mode uses one value each to control the individual sides of the drivetrain.
- Arcade Drive - This mode uses one value to control the throttle (speed along the X-axis) of the drivetrain and one for the rate of rotation.
- Curvature Drive - Also known as "Cheesy Drive" this is an alternate way of using one value to control throttle and one value for rotation. The rotation argument controls the curvature of the robot's path rather than its rate of heading change. This makes the robot more controllable at high speeds. Also handles the robot's quick turn functionality - "quick turn" overrides constant-curvature turning for turn-in-place maneuvers.

Tank Drive

The Tank Drive mode is used to control each side of the drivetrain independently (usually with an individual joystick axis controlling each). This example shows how to use the Y-axis of two separate joysticks to run the drivetrain in Tank mode. Construction of the objects has been omitted, for above for drivetrain construction and [here](#) for Joystick construction.

C++

```cpp
class Robot: public frc::TimedRobot
{
    //Object construction

    void TeleopPeriodic() override {
        myDrive.TankDrive(leftStick.GetY(), rightStick.GetY());
    }
}
```

Java

```java
public class RobotTemplate extends TimedRobot
{
    //Object construction
```
Arcade Drive

The Arcade Drive mode is used to control the drivetrain using speed/throttle and rotation rate. This is typically used either with two axes from a single joystick, or split across joysticks (often on a single gamepad) with the throttle coming from one stick and the rotation from another. This example shows how to use a single joystick with the Arcade mode. Construction of the objects has been omitted, for above for drivetrain construction and here for Joystick construction.

C++

```cpp
class Robot: public frc::TimedRobot
{
  //Object construction

  void TeleopPeriodic() override {
    myDrive.ArcadeDrive(driveStick.GetY(), driveStick.GetX());
  }
}
```

Java

```java
public class RobotTemplate extends TimedRobot
{
  //Object construction

  public void teleopPeriodic() {
    myDrive.arcadeDrive(driveStick.getY(), driveStick.getX());
  }
}
```
Curvature Drive

Like Arcade Drive, the Curvature Drive mode is used to control the drivetrain using speed/throttle and rotation rate. The difference is that the rotation control is attempting to control radius of curvature instead of rate of heading change. This mode also has a quick-turn parameter that is used to engage a sub-mode that allows for turning in place. This example shows how to use a single joystick with the Curvature mode. Construction of the objects has been omitted, for above for drivetrain construction and here for Joystick construction.

C++

```cpp
class Robot: public frc::TimedRobot
{
    //Object construction

    void TeleopPeriodic() override {
    }
}
```

Java

```java
public class RobotTemplate extends TimedRobot
{
    //Object construction

    public void teleopPeriodic() {
        myDrive.curvatureDrive(driveStick.getY(), driveStick.getX(), driveStick.getButton(1));
    }
}
```