```cpp
#include "drive.h"


//creates okapi chassis object
Drive::Drive() {
  chassis = ChassisControllerBuilder()
            .withMotors(
              {-TOP_LEFT_MOTOR, -LEFT_MIDDLE_MOTOR, -BOTTOM_LEFT_MOTOR},
              {TOP_RIGHT_MOTOR, RIGHT_MIDDLE_MOTOR, BOTTOM_RIGHT_MOTOR})
            .withDimensions(
              AbstractMotor::gearset::green,
                    ChassisScales({WHEELDIM, WHEELTRACK}, imev5GreenTPR))
            .withSensors(
              ADIEncoder( // left encoder
                  LEFT_TRACKING_WHEEL_TOP,
                  LEFT_TRACKING_WHEEL_BOTTOM

              ),
              ADIEncoder( // right encoder
                  RIGHT_TRACKING_WHEEL_TOP,
                  RIGHT_TRACKING_WHEEL_BOTTOM,
                  true
              )
          )
          .withOdometry(
            ChassisScales({ODOMWHEELDIM, ODOMTRACK}, quadEncoderTPR)
          )

        .buildOdometry();
        speedfactor = 1;
}




//returns X of odometry
double Drive::getX() {
  return chassis->getState().x.convert(inch);
}

//returns Y of odometry
double Drive::getY() {
  return chassis->getState().y.convert(inch);
}

//returns odometry heading
double Drive::getHeading() {
  return chassis->getState().theta.convert(degree);
}


//arcade move function for X drive (old)
void Drive::run(double forward, double strafe, double heading) {
  std::shared_ptr<okapi::XDriveModel> xDrive = std::static_pointer_cast<okapi::XDriveModel>
(chassis->getModel());
  if(forward+strafe+heading>1) {
    forward/=(forward+strafe+heading);
    strafe/=(forward+strafe+heading);
    heading/=(forward+strafe+heading);
  }
  printf("%f %f %f\n", strafe, forward, heading);
  xDrive->xArcade(strafe, forward, heading);
}

//arcade move function for tank drive
void Drive::runTankArcade(double forward, double turn) {
  chassis->getModel()->arcade(forward, turn);
}

//tank move function for tank drive
void Drive::runTank(double left, double right) {
  chassis->getModel()->tank(left, right);
}


//returns all of odometry state (x, y, and theta)
okapi::OdomState Drive::getState() {
  return chassis->getState();
}

//reverses orientation for driver
void Drive::reverseOrientation(int ori) {
  if(ori%2 == 1) {
    printf("REVERSED\n");
    speedfactor = -1;
  }
  else {
    speedfactor = 1;
  }
}

//sets brake mode of drive mode (if need to coast or hold)
void Drive::setMode(okapi::AbstractMotor::brakeMode brakeMode) {
  chassis->getModel()->setBrakeMode(brakeMode);
}

// double Drive::getEncoder() {
//   return enc.get();
// }
```