```
// Testing done by Evan Wwwnrneesridestanillia7ms
// For CSE 271
// Due 2/27/22
// Is used to Test the Car Class

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

class CarTester {
      Car tester = new Car(); //Tests empty Constructor
      @Test
      void testConstructorOne() {
            Car car1 = new Car();
            Boolean setProperly = true;
            if("" != car1.getOwner()) {
                  setProperly = false;
            }else if("" != car1.getMake()) {
                  setProperly = false;
            }else if("" != car1.getModel()) {
                  setProperly = false;
            }else if(0 != car1.getSpeed()) {
                  setProperly = false;
            }else if(1.0 != car1.getFuelLevel()) {
                  setProperly = false;
            }else if(2022 != car1.getYearModel()) {
                  setProperly = false;
            }else if(false != car1.isStart()) {
                  setProperly = false;
            }
            assertEquals(setProperly , true);

      }

      @Test
      void testConstructorTwo() {//Tests a constructor with some data
            Car car1 = new Car("Aidan", "Honda", "Civic", 1998);
            Boolean setProperly = true;
            if("Aidan" != car1.getOwner()) {
                  setProperly = false;
            }else if("Honda" != car1.getMake()) {
                  setProperly = false;
            }else if("Civic" != car1.getModel()) {
                  setProperly = false;
            }else if(0 != car1.getSpeed()) {
                  setProperly = false;
            }else if(1.0 != car1.getFuelLevel()) {
                  setProperly = false;
            }else if(1998 != car1.getYearModel()) {
                  setProperly = false;
            }else if(false != car1.isStart()) {
                  setProperly = false;
            }
            assertEquals(setProperly , true);
      }

      @Test
      void testConstructorThree() { //Tests Workhorse constructor
            Car car1 = new Car("Aidan", "Honda", "Civic", 1998, 0.75, 95, true);
```

```
		Boolean setProperly = true;
		if("Aidan" != car1.getOwner()) {
			setProperly = false;
		}else if("Honda" != car1.getMake()) {
			setProperly = false;
		}else if("Civic" != car1.getModel()) {
			setProperly = false;
		}else if(95 != car1.getSpeed()) {
			setProperly = false;
		}else if(0.75 != car1.getFuelLevel()) {
			setProperly = false;
		}else if(1998 != car1.getYearModel()) {
			setProperly = false;
		}else if(true != car1.isStart()) {
			setProperly = false;
		}
		assertEquals(setProperly , true);
	}

	@Test
	void testConstructorFour() {//test constructor that uses another car to set
the new cars values
		Car test = new Car("Aidan", "Honda", "Civic", 1998, 0.75, 95, true);
		Car car1 = new Car(test);
		Boolean setProperly = true;
		if("Aidan" != car1.getOwner()) {
			setProperly = false;
		}else if("Honda" != car1.getMake()) {
			setProperly = false;
		}else if("Civic" != car1.getModel()) {
			setProperly = false;
		}else if(95 != car1.getSpeed()) {
			setProperly = false;
		}else if(0.75 != car1.getFuelLevel()) {
			setProperly = false;
		}else if(1998 != car1.getYearModel()) {
			setProperly = false;
		}else if(true != car1.isStart()) {
			setProperly = false;
		}
		assertEquals(setProperly , true);
	}

	@Test
	void testSetOwner() {//checks to make sure that the setOwner method works
		String test = "Aidan Owens";
		try { //uses try catch to catch if user inputed a wrong argument when
setting the cars value
			tester.setOwner("Aidan Owens");
			assertEquals(tester.getOwner(), test);
		}catch(IllegalArgumentException e) {
			assertTrue(true);
		}

	}

	@Test
	void testSetMake() {//checks to make sure that the setMake method works
		String test = "Honda";
```

```
            try { //uses try catch to catch if user inputed a wrong argument when
setting the cars value
                  tester.setMake("Honda");
                  assertEquals(tester.getMake(), test);
            }catch(IllegalArgumentException e) {
                  assertTrue(true);
            }

      }

      @Test
      void testSetModel() {//checks to make sure that the setModel method works
            String test = "CRX";
            try { //uses try catch to catch if user inputed a wrong argument when
setting the cars value
                  tester.setModel("CRX");
                  assertEquals(tester.getModel(), test);
            }catch(IllegalArgumentException e) {
                  assertTrue(true);
            }

      }

      @Test
      void testSetYear() {//tests to see if the year being set is within the range
and if it is, it sees it if set it correctly
            int test = 2011;
            if(test >= 1885 && test <= 2022) {
            tester.setYearModel(2011);
            assertEquals(tester.getYearModel(), test);
            }else {  // but if it isnt in the range it tests to see if the illegal
argument exception is thrown and caught
                  try {
                        tester.setYearModel(test);
                         fail("Should have thrown an exception");
                  }catch(IllegalArgumentException e) {
                        assertTrue(true);
                  }
            }
      }

      @Test
      void testSetFuel() {//test to see if fuel is within allowed range, if it is
it checks if it is set properly
            double test = 0.75;
            if(test > 0.0 && test < 1.0) {
            tester.setFuelLevel(0.75);
            assertEquals(tester.getFuelLevel(), test);
            }else {// if it is not in the range it tests to see if the illegal
argument exception was thrown
                  try { //uses try catch to catch if user inputed a wrong argument
when setting the cars value
                        tester.setFuelLevel(test);
                         fail("Should have thrown an exception");
                  }catch(IllegalArgumentException e) {
                        assertTrue(true);
                  }
            }
      }
```

```
	@Test
	void testSetSpeed() {//test limits
		int test = 99;
		if(test >0 && test <250) {
		tester.setSpeed(99);
		assertEquals(tester.getSpeed(), test);
		}else {// if it is not in the range it tests to see if the illegal
argument exception was thrown
			try { //uses try catch to catch if user inputed a wrong argument
when setting the cars value
				tester.setSpeed(test);
				 fail("Should have thrown an exception");
			}catch(IllegalArgumentException e) {
				assertTrue(true);
			}
		}
	}

	@Test
	void testSetStart() {//checks to make sure that the setStart method works
		Boolean test = true;
		try { //uses try catch to catch if user inputed a wrong argument when
setting the cars value
			tester.setStart(true);
			assertEquals(tester.isStart(), test);
		}catch(IllegalArgumentException e) {
			assertTrue(true);
		}

	}

	@Test
	void testAccelerate() {//checks to make sure accelerate method works
regardless of if car is on or off
		tester.setStart(true);
		tester.setFuelLevel(1.0);
		tester.setSpeed(80);

		if(tester.isStart()) {
			tester.accelerate();
			int test = 84;
			assertEquals(tester.getSpeed(), test);
		}else {
			assertEquals(false, tester.isStart());
		}
	}


	@Test
	void testAccelerateMax() {//checks to make sure that accelerate works
properly if speed is max with more fuel
		int test = 250;
		tester.setSpeed(250);
		tester.setStart(true);
		tester.setFuelLevel(0.75);
		tester.accelerate();
		assertEquals(tester.getSpeed(), test);
	}
	@Test
	void testAccelerateFuelless() {//checks if the method accelerate works
properly if gas tank is empty/below 0.05
		tester.setStart(true);
		tester.setFuelLevel(0.02);
		tester.setSpeed(80);
		assertEquals(false, tester.accelerate());
	}

	@Test
	void testBrake() {//tests to make sure break method works regardless of if
car is on or off
		tester.setStart(true);
		tester.setSpeed(80);
		if(tester.isStart()) {
			tester.brake();
			int test = 77;
			assertEquals(tester.getSpeed(), test);
		}else {
			assertEquals(false, tester.isStart());
		}
	}

	@Test
	void testBrakeStopped() {//tests to make sure the brake method works if car
is below 3 speed
		tester.setStart(true);
		tester.setSpeed(2);
		tester.brake();
		assertEquals(0, tester.getSpeed());
	}


	@Test
	void testIsGasTankEmpty() {//checks to make sure the isGasTankEmpty method
properly returns the correct boolean value
		tester.setFuelLevel(0.7);
		Boolean test = false;
		assertEquals(tester.isGasTankEmpty(), test);
	}

	@Test
	void testSameOwner() {//tests to make sure the sameOwner method properly
returns the correct boolean value
		Car tester2 = new Car();
		String owner = "Aidan Owens";
		tester.setOwner(owner);
		tester2.setOwner(owner);
	    assertEquals(tester.getOwner(), tester2.getOwner());
	}

	@Test
	void testEquals() {//tests to make sure that the two cars have the same
values by checking that the equals method works properly and returns the correct
boolean value
		Car tester2 = new Car(tester);
		Boolean test = true;
		assertEquals(tester.equals(tester2), test);
```

```
      }

      @Test
      void testToString() {//tests to make sure the toString method works by
comparing if the string output is properly executed
            String owner = tester.getOwner();
            String make = tester.getMake();
            String model = tester.getModel();
            int yearModel = tester.getYearModel();
            int speed = tester.getSpeed();
            double fuelLevel = tester.getFuelLevel();
            String test = String.format("Owner: %s, Make: %s, Model: %s, Year: %d,
Speed: %d, Fuel Level: %.2f", owner, make, model, yearModel, speed, fuelLevel);

            assertEquals(tester.toString(), test);

      }
}
```