**public** **void** testFindIntersections()

{

Plane plane = **new** Plane(**new** Point3D(2,2,0), **new** Vector(-1, -1, 0));

// ============ Equivalence Partitions Tests ==============

Ray ray = **new** Ray(**new** Point3D(0.5,0.5, 0), **new** Vector(-1,-4, 0));

List intersectionPoints1 = plane.findIntersections(ray);

*assertEquals*("ERROR, there must be an intersection point", intersectionPoints1.~~size~~(), 1);

Ray ray2 = **new** Ray(**new** Point3D(-1, -2, 0), **new** Vector(-1, -2, 0));

List intersectionPoints2 = plane.findIntersections(ray2);

*assertEquals*("ERROR, there must not be an intersection point", intersectionPoints2, **null**);

// =============== Boundary Values Tests ==================

**try**

{

Ray ray3 = **new** Ray(**new** Point3D(-1, 1, 0), **new** Vector(-2, 2, 0));

List intersectionPoints3 = plane.findIntersections(ray3);

Ray ray5 = **new** Ray(**new** Point3D(0.25, 0.25, 0), **new** Vector(-1, -1, 0));

List intersectionPoints5 = plane.findIntersections(ray5);

//the start of the ray is not included

Ray ray6 = **new** Ray(**new** Point3D(0.5, 0.5, 0), **new** Vector(-1, -1, 0));

List intersectionPoints6 = plane.findIntersections(ray6);

//the start of the ray is not included

Ray ray8 = **new** Ray(**new** Point3D(0.75, 0.5, 0), **new** Vector(1.5, 1, 0));

List intersectionPoints8 = plane.findIntersections(ray8);

//the ray is included in the plane

Ray ray4 = **new** Ray(**new** Point3D(-1, 2, 0), **new** Vector(-2, 2, 0));

List intersectionPoints4 = plane.findIntersections(ray4);

}

**catch**(Exception e)

{

*fail*("ERROR, there are no intersection points, therefore it must return null");

}

**try**

{

Ray ray7 = **new** Ray(**new** Point3D(1, 1, 0), **new** Vector(-1, -1, 0));

List intersectionPoints7 = plane.findIntersections(ray7);

}

**catch**(Exception e)

{

*fail*("ERROR, there must be an intersection point");

}

}