Dynamics.java

```java
package myLib.rungeKutta;

import java.awt.geom.Point2D;
import java.util.List;
import myLib.utils.Utils;

/**
 *
 * @author tadaki
 */
abstract public class Dynamics {

    protected double t;//独立変数
    protected double y[];//従属変数
    protected double yInit[];//従属変数の初期値
    protected DifferentialEquation equation;//連立微分方程式
    protected final int numVar;//従属変数の数

    /**
     * コンストラクタ
     *
     * @param numVar 従属変数の数
     * @param initials 従属変数の初期値
     */
    public Dynamics(int numVar, double... initials) {
        this.numVar = numVar;
        y = new double[numVar];
        yInit = new double[numVar];
        for (int i = 0; i < initials.length; i++) {
            y[i] = initials[i];
            yInit[i] = initials[i];
        }
        t = 0;
    }

    /**
     * 全ての変数を再初期化する
     *
     * @return 初期化された従属変数の値
     */
    public double[] initialize() {
        for (int i = 0; i < numVar; i++) {
            y[i] = yInit[i];
        }
        t = 0;
        return y;
```

```
    }

    /**
     * 時間発展：1ステップ
     *
     * @param dt 時間
     * @return 新しい状態での従属変数の値
     */
    public double[] update(double dt) {
        double yy[] = RungeKutta.rk4(t, y, dt, equation);
        for (int i = 0; i < numVar; i++) {
            y[i] = yy[i];
        }
        t += dt;
        return yy;
    }

    /**
     * 時間発展
     *
     * @param tt 時間
     * @param nstep 時間ttをnstepに区切って時間発展
     * @return nstepの間の時間発展 (t, y[0])のリスト
     */
    public List<Point2D.Double> evolution(double tt, int nstep) {
        double yy[][] = RungeKutta.rkdumb(y, t, tt, nstep, equation);
        List<Point2D.Double> points = Utils.createList();
        double dt = tt / nstep;
        for (int i = 0; i < nstep; i++) {
            double ttt = i * dt;
            points.add(new Point2D.Double(ttt, yy[0][i]));
        }
        for (int i = 0; i < numVar; i++) {
            y[i] = yy[i][nstep - 1];
        }
        t += tt;
        return points;
    }
}
```