```java
import components.naturalnumber.NaturalNumber;

/**
 * Controller class.
 *
 * @author Put your name here
 */
public final class NNCalcController1 implements NNCalcController {

    /**
     * Model object.
     */
    private final NNCalcModel model;

    /**
     * View object.
     */
    private final NNCalcView view;

    /**
     * Useful constants.
     */
    private static final NaturalNumber TWO = new NaturalNumber2(2),
            INT_LIMIT = new NaturalNumber2(Integer.MAX_VALUE);

    /**
     * Updates this.view to display this.model, and to allow only operations
     * that are legal given this.model.
     *
     * @param model
     *            the model
     * @param view
     *            the view
     * @ensures [view has been updated to be consistent with model]
     */
    private static void updateViewToMatchModel(NNCalcModel model,
            NNCalcView view) {

        NaturalNumber input = model.bottom();
        NaturalNumber output = model.top();

        view.updateBottomDisplay(input);
        view.updateTopDisplay(output);
        /*
         * make sure the number would be smaller than the limit of the int
         */
        if (input.compareTo(INT_LIMIT) < 0) {
            view.updatePowerAllowed(true);
        } else {
            view.updatePowerAllowed(false);
        }
        /*
         * make sure the number which are subtracted would be correct
         */
        if (input.compareTo(output) < 0) {
            view.updateSubtractAllowed(true);
        } else {
            view.updateSubtractAllowed(false);
        }
        /*
         * make sure the number would be above 0
         */
        if (input.compareTo(TWO.newInstance()) > 0) {
            view.updateDivideAllowed(true);
        } else {
            view.updateDivideAllowed(false);
        }
        /*
         * make sure the number would be bigger than two and smaller than the
         * limit of the int
         */
        if (input.compareTo(TWO) >= 0 && input.compareTo(INT_LIMIT) <= 0) {
            view.updateRootAllowed(true);
        } else {
            view.updateRootAllowed(false);
        }
    }

    /**
     * Constructor.
     *
     * @param model
     *            model to connect to
     * @param view
     *            view to connect to
     */
    public NNCalcController1(NNCalcModel model, NNCalcView view) {
        this.model = model;
        this.view = view;
        updateViewToMatchModel(model, view);
    }

    @Override
    public void processClearEvent() {
        /*
         * Get alias to bottom from model
         */
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        bottom.clear();
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processSwapEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        NaturalNumber temp = top.newInstance();
        temp.transferFrom(top);
        top.transferFrom(bottom);
        bottom.transferFrom(temp);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processEnterEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        top.copyFrom(bottom);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processAddEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        bottom.add(top);
        top.clear();
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processSubtractEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        top.subtract(bottom);
        bottom.transferFrom(top);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processMultiplyEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        bottom.multiply(top);
        top.clear();
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processDivideEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        NaturalNumber tem = new NaturalNumber2();
        tem = top.divide(bottom);
        bottom.copyFrom(top);
        top.transferFrom(tem);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processPowerEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        top.power(bottom.toInt());
        bottom.transferFrom(top);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processRootEvent() {
        /*
         * Get aliases to top and bottom from model
         */
        NaturalNumber top = this.model.top();
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        top.root(bottom.toInt());
        bottom.transferFrom(top);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

    @Override
    public void processAddNewDigitEvent(int digit) {
        /*
         * Get aliases to bottom from model
         */
        NaturalNumber bottom = this.model.bottom();
        /*
         * Update model in response to this event
         */
        bottom.multiplyBy10(digit);
        /*
         * Update view to reflect changes in model
         */
        updateViewToMatchModel(this.model, this.view);
    }

}
```