

## Problem Set 2 Grading Policy

- Graded problem sets are due Monday, October 3 at noon.
- Make sure to post all the grades on MIT server by noon. Otherwise, you won't be able to post grades online.
- Return graded hardcopies to the TA in charge.

### 1. Basic

- All errors should be marked and explained. Any resulting point deductions should be noted, with reason, as close as possible to the error.
- The final score for any problem in a problem set should not be less than zero.

### 2. Late or Incomplete Submission

- The problem set score is the score a student gets after her/his problem set has been graded normally.
- If the timestamp for a problem set is after Friday at 11:05AM, deduct 30 points from the **problem set score**. If a problem set is late, please do mark it as late.
- If the timestamp for a problem set is after the following Monday at 11:05, the final grade is automatically 0. However, you should still fully correct the problem set to provide feedback.

### 3. When Programs Don't Compile

- If the compilation error seems like it was unintended, take off 10 points. In general, it should be easy to decide whether a compilation error is unintended or not. For instance, sometimes students accidentally delete semicolons after they are done testing and running their programs, but before their final submission. These are the cases we are trying to capture. In these cases, it must be evident that the trivial addition of a semicolon or the insertion of a missing bracket should make the program work.
- Otherwise, take off 30 points.
- If you are wondering whether a particular compilation error is unintended or not, contact the TA.

### 4. When Programs Don't Execute Correctly

- When you execute a student's program, follow the control of the program. If you find a problem, fit it into one of the case descriptions below. Take off as many points as indicated, correct the mistake, and continue the execution of the program.

- Do not double penalize student for the same mistake. Once you have corrected their initial mistake, keep executing the program to find more mistakes.
- If you find additional mistakes that DO NOT fall into the same case, take off points for these.
- You should have a good and consistent justification for each error student makes.
- Be consistent throughout the entire problem set.

## 5. Formatting & Commenting

- In general, if a solution is unnecessarily difficult to understand due to poor formatting, bad variable names, or lack of comments, you should deduct 5 points at most from the problem set score.

## 6. Specific Case Description for PS 1

- Please give FULL credit to any part that runs and produces the correct results.
- Students can write the Java code the way they like, and it may be different from the official solution.
- Note that, while the official solutions do not use methods, some of the students may have used done so. Do not penalize them for using methods.
- `Investment.java` (12 Points)
  - 2 points for each of the six (6) data members
- `Portfolio.java` (10 Points)
  - 2 points for each of the five (5) data members
- `ProblemSet2.java` (78 Points)
  - Exercise 1 (27 Points)
    - 12 points - 3 points each for instantiating the `Investment` and `Portfolio` objects and initializing the data members.
    - 12 points - calculation loop (do not penalize students who have done this exercise using a loop for each `Investment` object):
      - 3 points - Type 1 `Investment` (constant interest rate)
      - 5 points - Type 2 `Investment` (interest rate varies with age)
      - 4 points - Type 3 `Investment` (interest rate varies randomly)
    - 3 points – printing required information
  - Exercise 2 (51 Points)
    - 8 points – 2 points each for instantiating the `Investment` and `Portfolio` objects and initializing the data members
    - 40 points – calculation loop

- no money may be “lost” – all monies must end up in some account
  - there are many possible solutions; treat any which holds to the premise of the problem set as being valid.
- 3 points – printing required information