Hello Parent or Guardian,

I am excited to teach Computer Science: Program Your Own Role Playing Game. Below is a brief overview of my goals for your student:

- Learn fundamental programming concepts for imperative languages.
- Learn about the deeper mechanics involved in making an RPG or other game.
- Have fun.

Ultimately, I hope this class will create a source of intrinsic motivation for your student to learn more about computer science.

One may wonder why I chose to teach this class with a focus on gaming. When I was in high school I knew almost nothing about computer science, but my passion for gaming lead me to teach myself independently how to program. I see this passion for gaming as a positive motivator that can lead to a life of intellectual enrichment when funneled in the right direction. Furthermore, developing games can offer a programmer immediate satisfaction and results while covertly requiring one to learn math and logic at a high level. Thus, I believe the development of games has a lot to offer. For a good example of how games can both increase motivation and elicit hard work I would recommend:

E. Brunvand, "Games as motivation in computer design courses: I/O is the key," in Proceedings of the 42nd ACM technical symposium on Computer science education, ser. SIGCSE '11. New York, NY, USA: ACM, 2011, pp. 33–38. [Online].

Below one can find our agenda for the class. Each day there will be a homework assignment that should take 1 hour or less. I will make all class materials along with tutorials on where to get the software available online by the end of the class. If anyone has any questions or would like to see some of the homeworks, I can be reached by email (nakayama@iastate.edu).

Thank you,

Brian Nakayama

More about the instructor:
I work as a graduate assistant for the computer science department at Iowa State University and as a research assistant for the Laboratory for Nanoscale Self Assembly. Over the past year I was a teaching assistant for COMS 321: Computer Architectures, as well as a volunteer and judge for the K-12 Computational Thinking Competition at ISU. I wrote a thesis, "Universal Computation in the Prisoner's Dilemma Game", for my BS with Honors from Regis University in 2013. I will be publishing a shorter version of my thesis for the Unconventional Computation and Natural Computation 2014 conference in London, Ontario, where I will also give a short talk this summer.

Agenda
Day 1:
- 20 minutes - questionaire on name, why they're interested, and programming experience.
- 40 minutes - explain what the class is, what Java is and what the class shall teach.
- 30 minutes - do Hello World and compile using the terminal.
- 30 minutes - start slightly harder Hello World example that takes argument from the terminal.
- 1 hour short lecture on different types of variables, what they represent, and how they are used.
- --break--
- 1 hour - continue working on the extended Hello World program. Introduce "if" statements. Review print and println methods and special characters.
- 2 hours - Introduce scanner class, as well as basic arithmetic for numbers and different number types. Create a basic text based RPG to practice basic understanding.

Day 2:
- Depending on the student's level of understanding student may be given different advanced tasks that will help students learn about a different area in Computer Science while doing similar programming activities with the class.
- 20 minutes - go over homework.
- 1 hour - introduce arrays, loops, and switch-case statements. Do quizGame example.
- 1 hour 40 minutes - continue to develop RPG by giving it a map.
- --break--
- 1 hour - lecture on methods and references.
- 1 hour - practice using methods, and also refactor RPG into methods.
- 30 minute - lecture on recursion.
- 30 minute - factorial example.

Day 3:

- 20 minutes - go over homework.
- 30 minutes - switch over to Eclipse IDE. Explain basic functionalities of the IDE.
- 30 minutes - import files into Eclipse and practice compiling.
- 40 minutes - introduce File I/O, and exception handling.
- 1 hour - practice File I/O and diary example.
- --break--
- 1 hour - implement File I/O to save/open maps and games on RPG.
- 1 hour - lecture on scope of variables: private, protected, default, and public.
- 1 hour - if not done so yet, create more than one class and practice importing.

Day 4:
- 20 minutes - go over homework.
- 1 hour - graphics demo, and presentation on advanced computer science topic.
- 40 minutes - introduction to basic GUI.
- 1 hour - implement a GUI for the diary example.
- --break--
- 1 hour - introduction to simple game libraries and 2d rendering.

- 1 hour - lecture on other programming languages and their uses. Students may work on concept for their game.
- 1 hour - implement basic 2d graphics for rpg.

Day 5:

- 20 minutes - go over homework.
- 40 minutes - introduction to bitwise operators and ternary operators.
- 40 minutes - in class exercise.
- 1 hour 20 minutes - continue working on RPG.
- --break--
- 1 hour 30 minutes - continue working on RPG.
- 30 minutes - lecture on how to create an executable jar so that they can show their final project to friends and family.
- 30 minutes - introduction to some OO concepts.
- 30 minutes - Hand out review sheet. Thank the students for their participation.

Day 6:

- Help students set up their demo.