Names: Mohammad Khan & Kevin Cao & Evelyn Zheng & Aryan Patel & Xian Jun An & Noakai Aronesty

Design of Simulation:

- **Animal.java:**
  - The attributes are the **current position**(to keep track of where each animal is), **maximum position**(to make sure the animal doesn't go out of the range of the river), and **previous position**(if an animal has to go back to its original position after attempting to move)
  - It has **two constructors**, one for when the position in the river is known(most of the time) and one for when it is not known(example in Simulator.java:54)
  - **move()** updates the animal's position with equal probability of moving left, right, or staying in the same position. **getPos()** is used to access the position because the variable is private.
  - **goBack()** sets an animal's position to the position they were before they attempted to move. (used when two of the same animal occupy the same cell)
  - **create()** creates an animal of the same type in a new position in the array (used when two of the same animal meet in the same cell)
- **Bear.java/Fish.java**
  - The **two constructors** accept the same values as the constructors in Animal.java and it sends this information to the Animal class because the species of an animal does not change their current position or the max river length so it can be generalized.
  - **equals()** solely checks if they are the same species of animal, position doesn't matter.
  - **toString()** just prints out 'F' or 'B' depending on the species (used in the printing function)
- **Simulator.java**
  - **createRiver()** randomizes the positions using a technique that we learned in the first semester and then adds animals to the ArrayList at these random positions.
  - **printRiver()** represents the river as a string and also counts the current population of each species. ("|" was added to show the boundaries of the river more clearly).
  - **updateRiver()** does casework to update the position of each animal and returns a new array with these updated positions. (Casework is explained in the comments in updateRiver()).
  - The main function first accepts the arguments and checks to see if the **combined population size(cps) <= the river length(rl)**, otherwise it throws an error because if the simulation was run with cps > rl there would be animals missing from the simulation because there is no space. It then initializes the river using createRiver(), and runs iterations of printing and updating the river using the appropriate functions.