OOP - Constructors & Mutators

Constructors

- Job = Instantiate an object

- Can pass in variables / set variables
  - BUT is not required to

Constructors

- Examples *with passing in variables*

Robot.h

```
class Robot {
        public:
                robot(string name, int health);
        private:
                string name;
                int health;
}
```

Robot.cpp

```
Robot::Robot(string name, int health) {
        name = name;
        health = health;
}

int main() {
        Robot r1(“Terminator”, 1000);
        return 0;
}
```

Constructors

*Constructors that pass in variables MUST pass in EXACTLY that number of variables, in the correct order, to be instantiated

- Examples *with passing in variables*

Robot.h

```
class Robot {
        public:
                robot(string name, int health);
        private:
                string name;
                int health;
}
```

Robot.cpp

```
Robot::Robot(string name, int health) {
        name = name;
        health = health;
}

int main() {
        Robot r1(“Terminator”, 1000);
        return 0;
}
```

Constructors

- Examples *without variables*

Robot.h

```
class Robot {
        public:
                robot();
        private:
                string name;
                int health;
}
```

Robot.cpp

```
Robot::Robot() {
        name = “Terminator”;
        health = 1000;
}

int main() {
        Robot r1;
        return 0;
}
```

Constructors

- Examples *without variables*

The variables can be hard coded in the constructor...

Robot.h

```
class Robot {
        public:
                robot();
        private:
                string name;
                int health;
}
```

Constructors can have NO variables

Robot.cpp

```
Robot::Robot() {
        name = “Terminator”;
        health = 1000;
}

int main() {
        Robot r1;
        return 0;
}
```

Constructors

- Examples *without variables*

Robot.h

```
class Robot {
        public:
                robot();
        private:
                string name;
                int health;
}
```

Constructors can have NO variables

Robot.cpp

```
Robot::Robot() {
}

int main() {
        Robot r1;
        return 0;
}
```

The variables can be set up in the constructor…
...or not at all! (the constructor is still necessary, however)

Constructors

- Examples *without variables*

Robot.h

```
class Robot {
        public:
                robot();
        private:
                string name;
                int health;
}
```

Constructors can have NO variables

Robot.cpp

```
Robot::Robot() {
        name = “Terminator”;
        health = 1000;
}

int main() {
        Robot r1;
        return 0;
}
```

The variables can be set up in the constructor.

If no variables are passed in, an object is created like a simple variable (no parentheses, etc…)

Constructors

- Back to this example: How does this object get to have values?!?

Robot.h

```
class Robot {
        public:
                robot();
        private:
                string name;
                int health;
}
```

Constructors can have NO variables

Robot.cpp

```
Robot::Robot() {
}

int main() {
        Robot r1;
        return 0;
}
```

The variables can be set up in the constructor…
...or not at all! (the constructor is still necessary, however)

Mutators

- Mutators = methods that *change* the value of a private variable
- Example:

```
robot::setName() {
   cout << “Enter a name: ”;
   cin >> name;
}
```

Mutators

- Mutators = methods that *change* the value of a private variable
- Can also be hard-coded

```
robot::setName() {
   name = “Terminator”;
}
```

Mutators

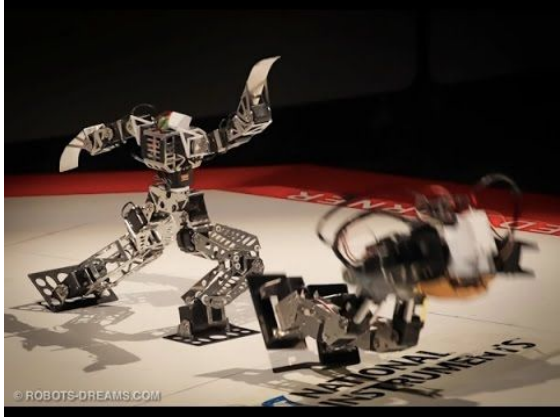
- Mutators = methods that *change* the value of a private variable
- Can also be set using calculations

```
robot::setHealth() {
   health = rand() % 20;
}
```

Mutators

- All variables in a class must be set one of two ways:
  - In a constructor (passed in or hard-coded)
  - Through a mutator method

Robot Class changes



- The name and health should NOT be passed into the constructor
  - Should be set using methods named setName() and setHealth();

- Create two robots (each with a different name and health, scanned in by the user) and print out information for each