Extended Review From Last Time

- p.getName() has value "W. White"
- p.pay(250.00) sets owes to 0

Class versus Object

Anatomy of a declaration + assignment statement:

- int x = 2;
- Type Variable Value
- Point3d p = new Point3d();

The Value null

- You can declare a class variable w/o using new:
  - Example: Point3d var3;
  - Value in variable is null
    - null: Absence of a name
    - var3.getX() gives error!
    - There is no name in var3
    - Does not know which Point3d to access
    - NullPointerException

Class Definition

- Describes the format of a folder (instance, object) of the class.

Field: A Variable in each Folder of a Class

- Methods are the key doers

We Write Programs to Do Things

- Method Definition
  - Defines what method does
  - Method Call
  - Command to do the method

Field: A Variable in each Folder of a Class

- Methods are the key doers

We Write Programs to Do Things

- Method Definition
  - Defines what method does
  - Method Call
  - Command to do the method
### Getter and Setter Methods

```java
/** Yields: worker’s last name*/
public String getName() {
    return lname;
}
/** Set worker’s last name to n
* Cannot be null; can be “” */
public void setName(String n) {
    lname = n;
}
/** Yields: last 4 SSN digits, as int *
* Try writing it yourself.
* Full code on website
*/
public int getName() {
    return ssn;
}
```

---

### Why Getters and Setters?

#### Setters
- Protect field invariants
- Example:
  ```java
  public void setName(String n) {
      lname = n;
      if (n == null) {
          lname = "";
      }
  }
  ```

#### Getters
- Allow “read”, not “write”
- Example:
  ```java
  public int getName() {
      return ssn;
  }
  ```

---

### Initializing the Fields of an Object (Folder)

- Creating a new Worker is now a multi-step process:
  ```java
  Worker w = new Worker();
  w.setName("White");
  ...
  ```
- We would like to be able to use something like
  ```java
  Worker w = new Worker("White", 1, null);
  ```
- Need a special kind of method: the constructor

---

### Example Constructor

```java
/** Constructor: an instance with last
* name n (can’t be null, can be ""),
* SSN s (an int in 0.999999999), and
* boss b (null if none)
*/
public Worker(String n, int s, Worker b) {
    name = n;
    ssn = s;
    boss = b;
    name = name;
    ssn = s;
    boss = b;
}
```

---

### How Methods Work

- **Example:**
  - Get object (folder) name
    ```java
    var1.getX()
    ```
  - Searches class (file drawer)
    ```java
    var1.getName()
    ```
  - Executes commands inside the method on that object
    ```java
    var1.setX(8.2);
    ```
  - Methods apply to the object (folder), not the variable!
    ```java
    var2.setX(8.2);
    ```
  - Makes `var3.getX() == 8.2`
    ```java
    var3.x = 8.2
    ```

---

### Initializing the Fields of an Object (Folder)

- Creating a new Worker is now a multi-step process:
  - Worker w = new Worker();
  - w.setName(“White”);
  - ...
- We would like to be able to use something like
  ```java
  Worker w = new Worker(“White”, 1, null);
  ```
- Need a special kind of method: the constructor

---

### Purpose of the Constructor

- Initialize the fields of a newly created object
- Make sure that the invariants are true
- The SSN to 0000000001, and the boss to null.
- Need a special kind of method: the constructor

---

### Memorize This!

Write it down several times.