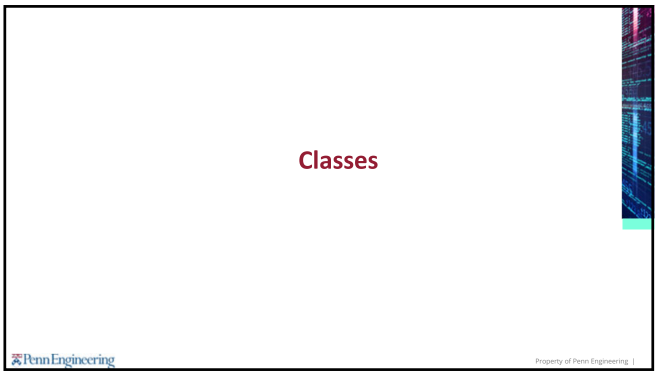
Classes

Brandon Krakowsky

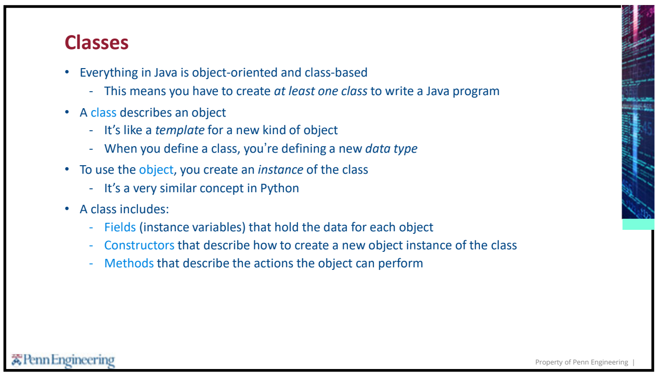
Penn Engineering

1



Classes

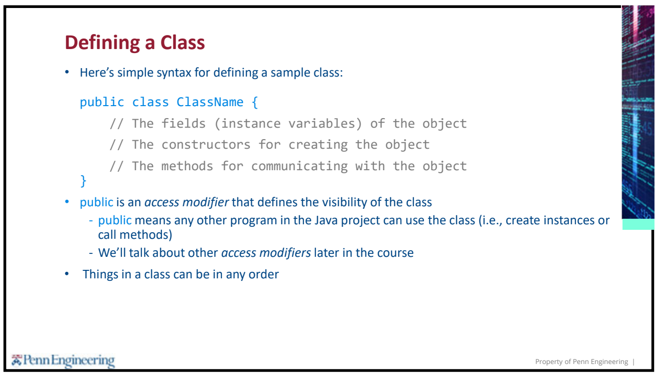
2



Classes

- Everything in Java is object-oriented and class-based
  - This means you have to create *at least one class* to write a Java program
- A class describes an object
  - It's like a *template* for a new kind of object
  - When you define a class, you're defining a new *data type*
- To use the object, you create an *instance* of the class
  - It's a very similar concept in Python
- A class includes:
  - Fields (instance variables) that hold the data for each object
  - Constructors that describe how to create a new object instance of the class
  - Methods that describe the actions the object can perform

3



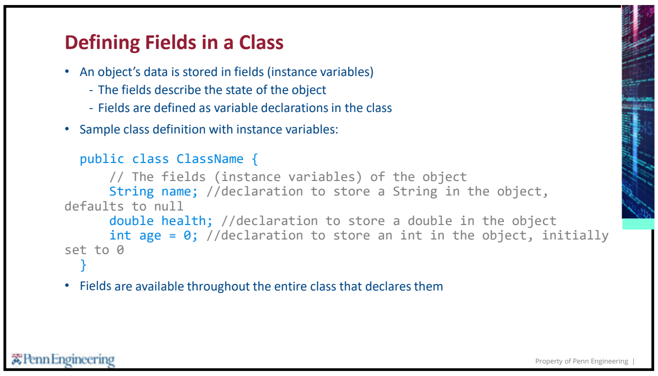
Defining a Class

- Here's simple syntax for defining a sample class:

```
public class ClassName {
    // The fields (instance variables) of the object
    // The constructors for creating the object
    // The methods for communicating with the object
}
```

- public is an *access modifier* that defines the visibility of the class
  - public means any other program in the Java project can use the class (i.e., create instances or call methods)
  - We'll talk about other *access modifiers* later in the course
- Things in a class can be in any order

4



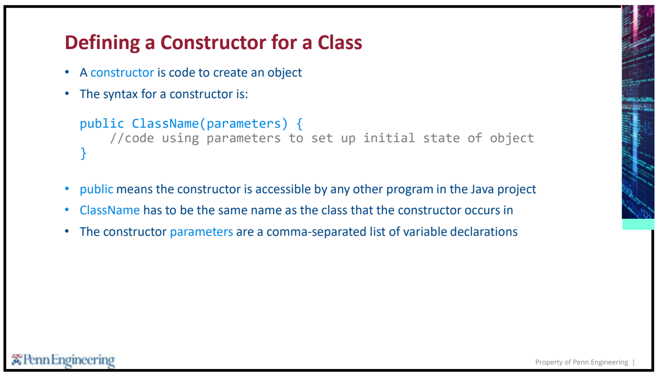
Defining Fields in a Class

- An object's data is stored in fields (instance variables)
  - The fields describe the state of the object
  - Fields are defined as variable declarations in the class
- Sample class definition with instance variables:

```
  public class ClassName {
      // The fields (instance variables) of the object
      String name; //declaration to store a String in the object,
defaults to null
      double health; //declaration to store a double in the object
      int age = 0; //declaration to store an int in the object, initially
set to 0
  }
```

- Fields are available throughout the entire class that declares them

5



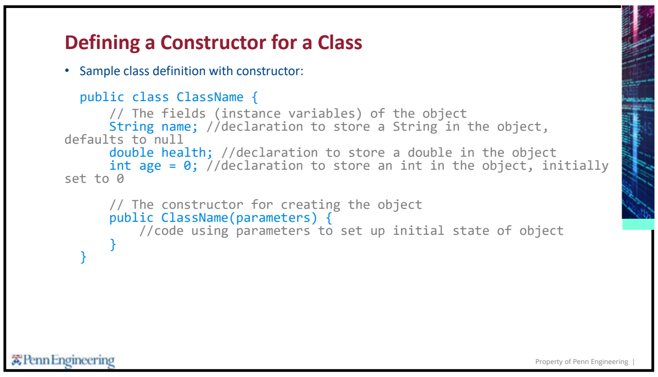
Defining a Constructor for a Class

- A constructor is code to create an object
- The syntax for a constructor is:

```
public ClassName(parameters) {
    //code using parameters to set up initial state of object
}
```

- public means the constructor is accessible by any other program in the Java project
- ClassName has to be the same name as the class that the constructor occurs in
- The constructor parameters are a comma-separated list of variable declarations

6



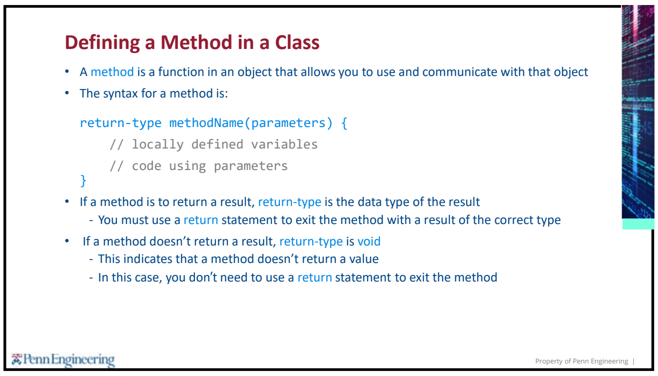
Defining a Constructor for a Class

- Sample class definition with constructor:

```
public class ClassName {
     // The fields (instance variables) of the object
     String name; //declaration to store a String in the object, 
defaults to null
     double health; //declaration to store a double in the object
     int age = 0; //declaration to store an int in the object, initially 
set to 0

     // The constructor for creating the object
     public ClassName(parameters) {
          //code using parameters to set up initial state of object
     }
}
```

Penn Engineering   Property of Penn Engineering |

7



Defining a Method in a Class

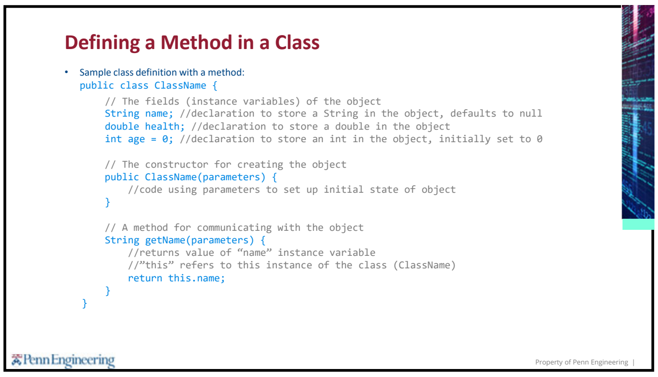
- A method is a function in an object that allows you to use and communicate with that object
- The syntax for a method is:

```
return-type methodName(parameters) {
     // locally defined variables
     // code using parameters
}
```

- If a method is to return a result, return-type is the data type of the result
  - You must use a return statement to exit the method with a result of the correct type
- If a method doesn't return a result, return-type is void
  - This indicates that a method doesn't return a value
  - In this case, you don't need to use a return statement to exit the method

Penn Engineering   Property of Penn Engineering |

8



Defining a Method in a Class

- Sample class definition with a method:

```
public class ClassName {
    // The fields (instance variables) of the object
    String name; //declaration to store a String in the object, defaults to null
    double health; //declaration to store a double in the object
    int age = 0; //declaration to store an int in the object, initially set to 0

    // The constructor for creating the object
    public ClassName(parameters) {
        //code using parameters to set up initial state of object
    }

    // A method for communicating with the object
    String getName(parameters) {
        //returns value of “name” instance variable
        //”this” refers to this instance of the class (ClassName)
        return this.name;
    }
}
```

Penn Engineering   Property of Penn Engineering |

9

Creating an Instance of a Class

- To use a class, you create an instance of the object by calling its constructor and using the keyword new
- Here's syntax to define a class and to create an instance:

```
public class ClassName {

    public ClassName(par1, … parN) {
        //code using parameters to set up initial state of object
    }

    public static void main(String[] args) {
        //create instance of ClassName
        ClassName c = new ClassName(arg1, …, argN);
    }
}
```

- new creates a new instance of the object

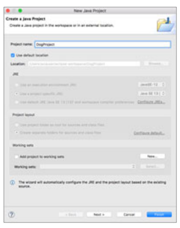
10

Dog Project

11

Dog Project

- In Eclipse, go to "File" → "New" → "Project"



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: DogProject

[✓] Use default location

JRE

Project layout

Working sets

[ ] Add project to working sets

Working sets:

The wizard will automatically configure the JRE and the project layout based on the existing source.

Provide a Project name
- Project names should be capitalized

Use the default location

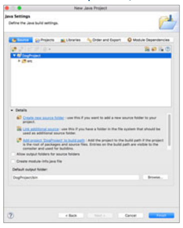
Use the default JRE and project layout

Click "Next"

12

Dog Project

- Define the compilation/build settings



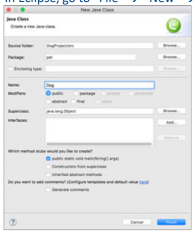
Make sure Create module-info.java file IS NOT checked

Use the default output folder

Click "Finish"

Dog Class

- In Eclipse, go to "File" → "New" → "Class"



Provide a Package name
- Package names should not be capitalized

Provide a Class name
- Class names should be capitalized

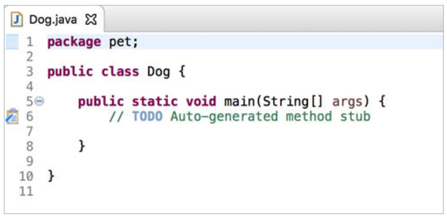
Make sure public static void main(String[] args) IS checked

Make sure Inherited abstract methods IS NOT checked

Click "Finish"

Dog Class

- The entry point of your Dog program is the *main* method

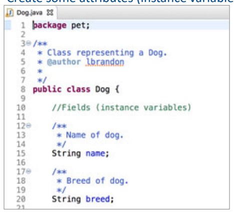


Dog.java

```
package pet;

public class Dog {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Dog Class – Instance Variables

- Add javadocs to your classes, variables, and methods as you write your program
- Create some attributes (instance variables) for your Dog

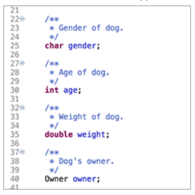


Dog.java

```java
package pet;

/**
 * Class representing a Dog.
 * @author lbrandon
 *
 */
public class Dog {

    //Fields (instance variables)

    /**
     * Name of dog.
     */
    String name;

    /**
     * Breed of dog.
     */
    String breed;
```

Dog Class – Instance Variables

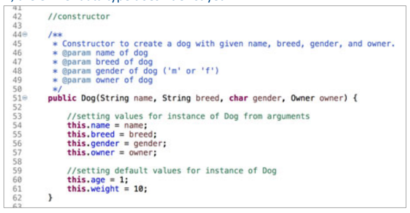
- Add javadocs to your classes, variables, and methods as you write your program
- Create some attributes (instance variables) for your Dog
  - The *Owner* data type doesn't exist yet. We'll create it as another class soon.



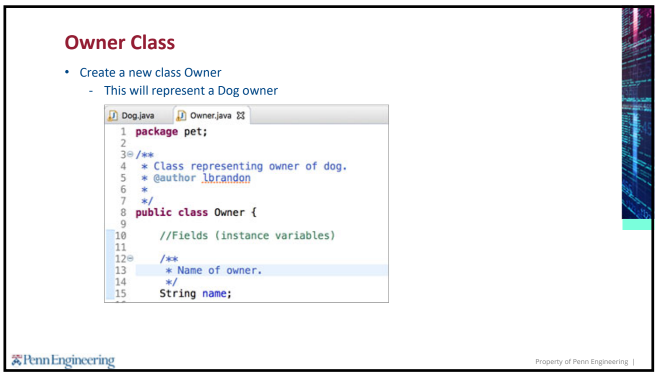
```java

    /**
     * Gender of dog.
     */
    char gender;

    /**
     * Age of dog.
     */
    int age;

    /**
     * Weight of dog.
     */
    double weight;

    /**
     * Dog's owner.
     */
    Owner owner;
```

Dog Class - Constructor

- Create a constructor for your object
  - Again, the *Owner* data type doesn't exist yet.



```java
    //constructor

    /**
     * Constructor to create a dog with given name, breed, gender, and owner.
     * @param name of dog
     * @param breed of dog
     * @param gender of dog ('m' or 'f')
     * @param owner of dog
     */
    public Dog(String name, String breed, char gender, Owner owner) {

        //setting values for instance of Dog from arguments
        this.name = name;
        this.breed = breed;
        this.gender = gender;
        this.owner = owner;

        //setting default values for instance of Dog
        this.age = 1;
        this.weight = 10;
    }
```



Owner Class

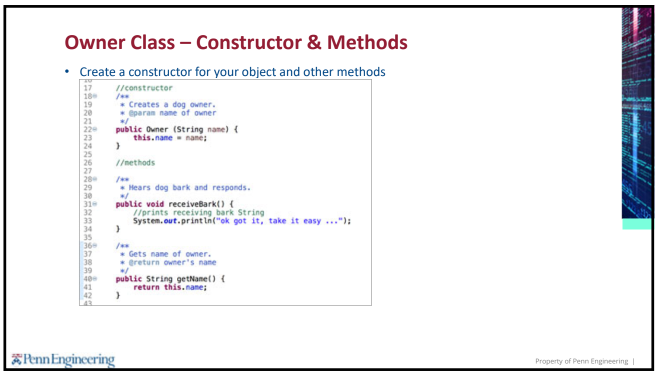
- Create a new class Owner
  - This will represent a Dog owner

Dog.java Owner.java

```java
package pet;

/**
 * Class representing owner of dog.
 * @author lbrandon
 *
 */
public class Owner {

    //Fields (instance variables)

    /**
     * Name of owner.
     */
    String name;
```

Penn Engineering   Property of Penn Engineering |

19



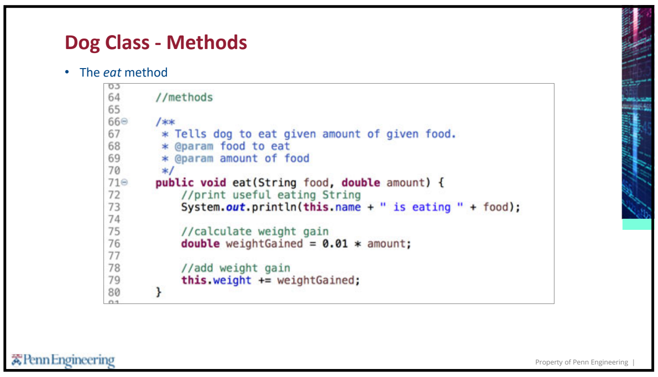
Owner Class – Constructor & Methods

- Create a constructor for your object and other methods

```java
    //constructor
    /**
     * Creates a dog owner.
     * @param name of owner
     */
    public Owner (String name) {
        this.name = name;
    }

    //methods

    /**
     * Hears dog bark and responds.
     */
    public void receiveBark() {
        //prints receiving bark String
        System.out.println("ok got it, take it easy ...");
    }

    /**
     * Gets name of owner.
     * @return owner's name
     */
    public String getName() {
        return this.name;
    }
```

Penn Engineering   Property of Penn Engineering |

20



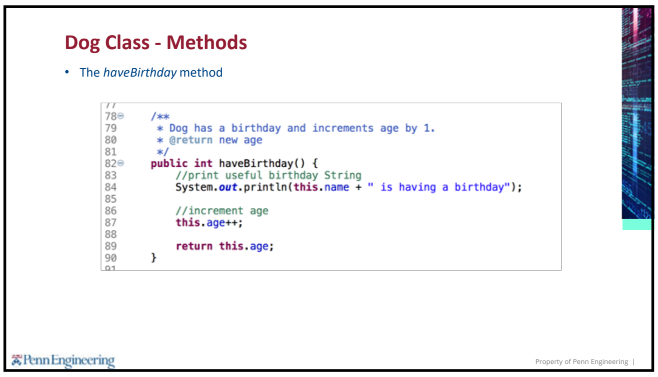
Dog Class - Methods

- The *eat* method

```java
    //methods

    /**
     * Tells dog to eat given amount of given food.
     * @param food to eat
     * @param amount of food
     */
    public void eat(String food, double amount) {
        //print useful eating String
        System.out.println(this.name + " is eating " + food);

        //calculate weight gain
        double weightGained = 0.01 * amount;

        //add weight gain
        this.weight += weightGained;
    }
```

Penn Engineering   Property of Penn Engineering |

21

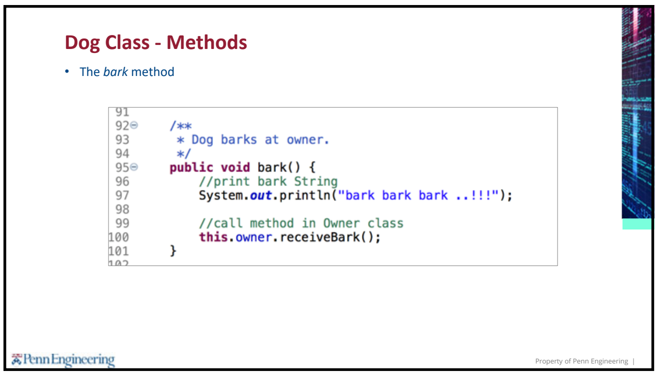


Dog Class - Methods

- The *haveBirthday* method

```java
    /**
     * Dog has a birthday and increments age by 1.
     * @return new age
     */
    public int haveBirthday() {
        //print useful birthday String
        System.out.println(this.name + " is having a birthday");

        //increment age
        this.age++;

        return this.age;
    }
```

22

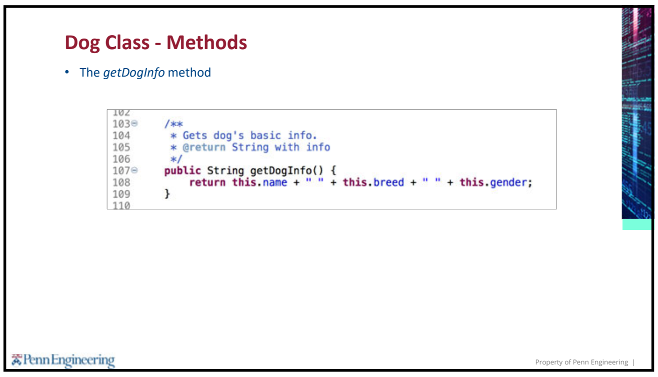


Dog Class - Methods

- The *bark* method

```java
    /**
     * Dog barks at owner.
     */
    public void bark() {
        //print bark String
        System.out.println("bark bark bark ..!!!");

        //call method in Owner class
        this.owner.receiveBark();
    }
```

23

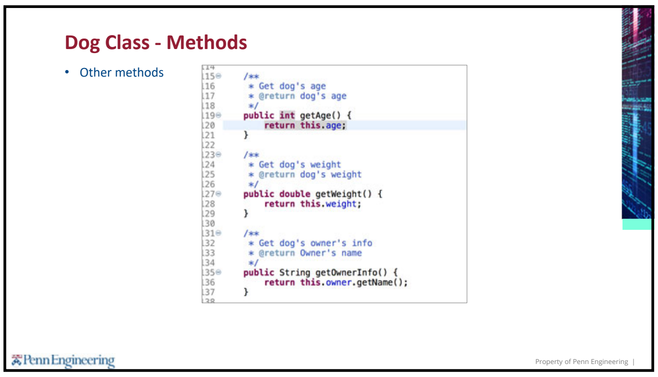


Dog Class - Methods

- The *getDogInfo* method

```java
    /**
     * Gets dog's basic info.
     * @return String with info
     */
    public String getDogInfo() {
        return this.name + " " + this.breed + " " + this.gender;
    }
```

24



Dog Class - Methods

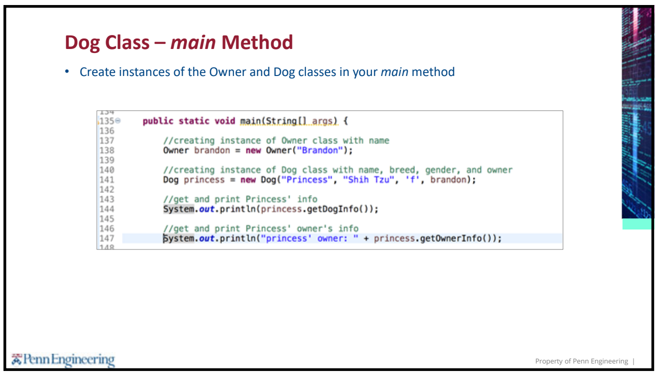
- Other methods

```java
    /**
     * Get dog's age
     * @return dog's age
     */
    public int getAge() {
        return this.age;
    }

    /**
     * Get dog's weight
     * @return dog's weight
     */
    public double getWeight() {
        return this.weight;
    }

    /**
     * Get dog's owner's info
     * @return Owner's name
     */
    public String getOwnerInfo() {
        return this.owner.getName();
    }
```

Penn Engineering

Property of Penn Engineering |

25



Dog Class – *main* Method

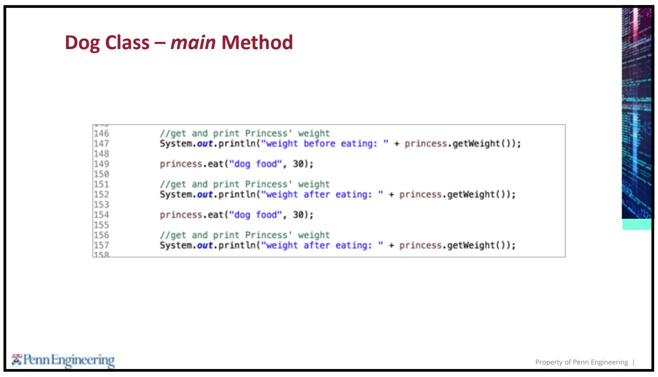
- Create instances of the Owner and Dog classes in your *main* method

```java
    public static void main(String[] args) {

        //creating instance of Owner class with name
        Owner brandon = new Owner("Brandon");

        //creating instance of Dog class with name, breed, gender, and owner
        Dog princess = new Dog("Princess", "Shih Tzu", 'f', brandon);

        //get and print Princess' info
        System.out.println(princess.getDogInfo());

        //get and print Princess' owner's info
        System.out.println("princess' owner: " + princess.getOwnerInfo());
```

Penn Engineering

Property of Penn Engineering |

26



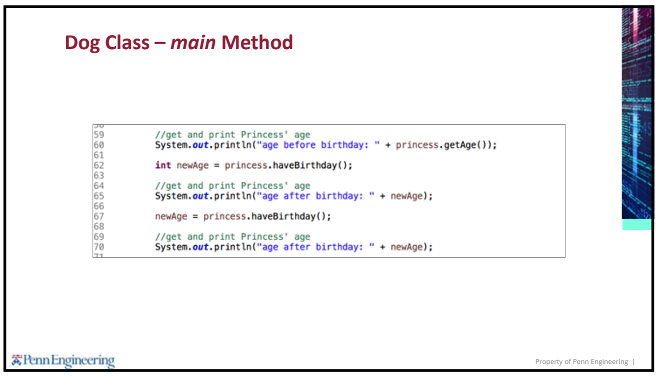
Dog Class – *main* Method

```java
        //get and print Princess' weight
        System.out.println("weight before eating: " + princess.getWeight());

        princess.eat("dog food", 30);

        //get and print Princess' weight
        System.out.println("weight after eating: " + princess.getWeight());

        princess.eat("dog food", 30);

        //get and print Princess' weight
        System.out.println("weight after eating: " + princess.getWeight());
```

Penn Engineering

Property of Penn Engineering |

27



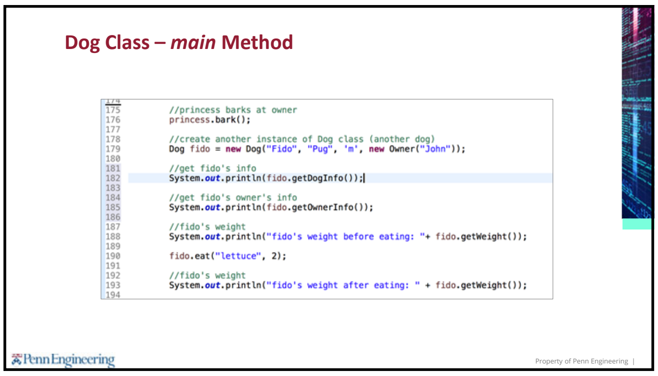
Dog Class – *main* Method

```
        //get and print Princess' age
        System.out.println("age before birthday: " + princess.getAge());

        int newAge = princess.haveBirthday();

        //get and print Princess' age
        System.out.println("age after birthday: " + newAge);

        newAge = princess.haveBirthday();

        //get and print Princess' age
        System.out.println("age after birthday: " + newAge);
```

Penn Engineering

Property of Penn Engineering |

28



Dog Class – *main* Method

```
        //princess barks at owner
        princess.bark();

        //create another instance of Dog class (another dog)
        Dog fido = new Dog("Fido", "Pug", 'm', new Owner("John"));

        //get fido's info
        System.out.println(fido.getDogInfo());

        //get fido's owner's info
        System.out.println(fido.getOwnerInfo());

        //fido's weight
        System.out.println("fido's weight before eating: "+ fido.getWeight());

        fido.eat("lettuce", 2);

        //fido's weight
        System.out.println("fido's weight after eating: " + fido.getWeight());
```

Penn Engineering

Property of Penn Engineering |

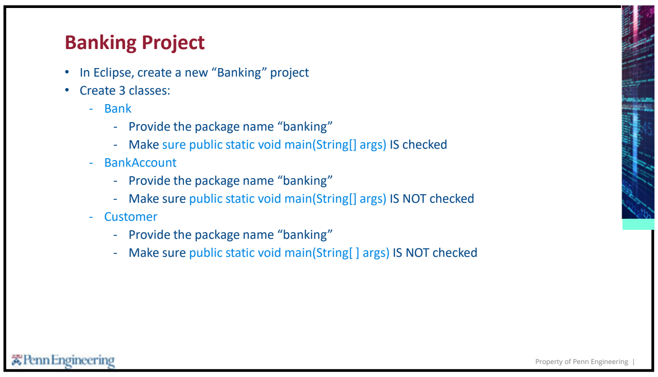
29

Banking Project

Penn Engineering

Property of Penn Engineering |

30

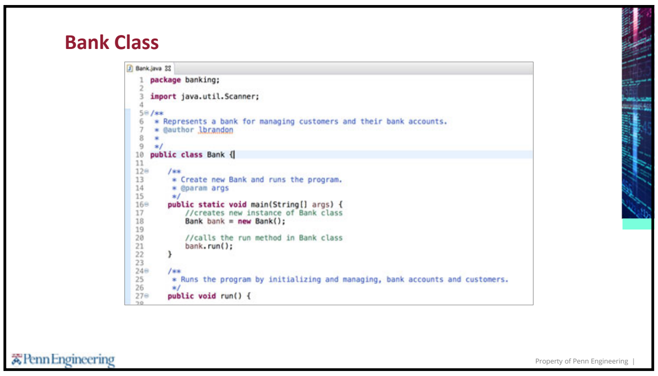


Banking Project

- In Eclipse, create a new "Banking" project
- Create 3 classes:
  - Bank
    - Provide the package name "banking"
    - Make sure public static void main(String[] args) IS checked
  - BankAccount
    - Provide the package name "banking"
    - Make sure public static void main(String[] args) IS NOT checked
  - Customer
    - Provide the package name "banking"
    - Make sure public static void main(String[ ] args) IS NOT checked

Penn Engineering

31



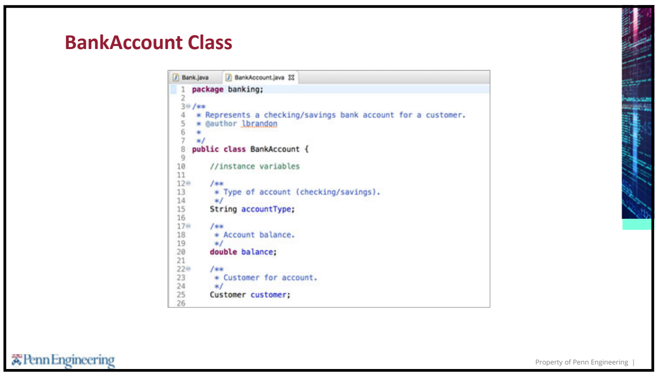
Bank Class

Bank.java

```java
package banking;

import java.util.Scanner;

/**
 * Represents a bank for managing customers and their bank accounts.
 * @author lbrandon
 *
 */
public class Bank {

    /**
     * Create new Bank and runs the program.
     * @param args
     */
    public static void main(String[] args) {
        //creates new instance of Bank class
        Bank bank = new Bank();

        //calls the run method in Bank class
        bank.run();
    }

    /**
     * Runs the program by initializing and managing, bank accounts and customers.
     */
    public void run() {
```

Penn Engineering

32



BankAccount Class

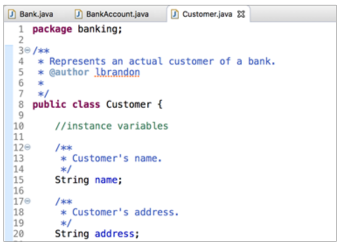
Bank.java BankAccount.java

```java
package banking;

/**
 * Represents a checking/savings bank account for a customer.
 * @author lbrandon
 *
 */
public class BankAccount {

    //instance variables

    /**
     * Type of account (checking/savings).
     */
    String accountType;

    /**
     * Account balance.
     */
    double balance;

    /**
     * Customer for account.
     */
    Customer customer;

```

Penn Engineering

33

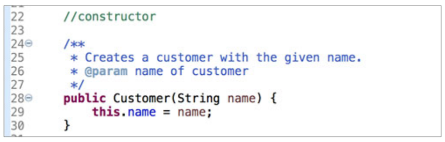
Customer Class



```java
package banking;

/**
 * Represents an actual customer of a bank.
 * @author lbrandon
 *
 */
public class Customer {

    //instance variables

    /**
     * Customer's name.
     */
    String name;

    /**
     * Customer's address.
     */
    String address;
```

34

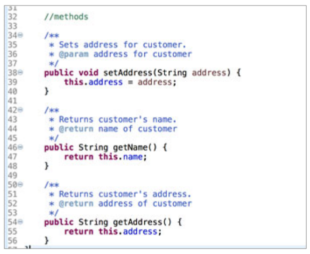
Customer Class



```java
    //constructor

    /**
     * Creates a customer with the given name.
     * @param name of customer
     */
    public Customer(String name) {
        this.name = name;
    }
```

35

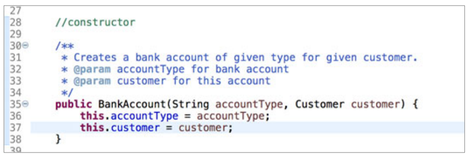
Customer Class



```java
    //methods

    /**
     * Sets address for customer.
     * @param address for customer
     */
    public void setAddress(String address) {
        this.address = address;
    }

    /**
     * Returns customer's name.
     * @return name of customer
     */
    public String getName() {
        return this.name;
    }

    /**
     * Returns customer's address.
     * @return address of customer
     */
    public String getAddress() {
        return this.address;
    }
```

36

BankAccount Class

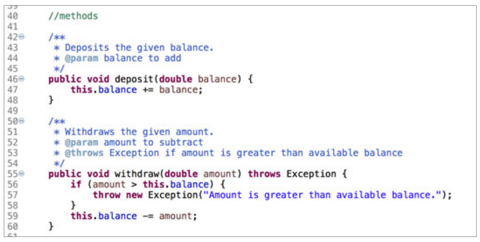


```java
    //constructor

    /**
     * Creates a bank account of given type for given customer.
     * @param accountType for bank account
     * @param customer for this account
     */
    public BankAccount(String accountType, Customer customer) {
        this.accountType = accountType;
        this.customer = customer;
    }
```

Penn Engineering   Property of Penn Engineering |

37

BankAccount Class

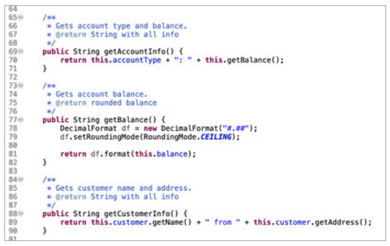


```java
    //methods

    /**
     * Deposits the given balance.
     * @param balance to add
     */
    public void deposit(double balance) {
        this.balance += balance;
    }

    /**
     * Withdraws the given amount.
     * @param amount to subtract
     * @throws Exception if amount is greater than available balance
     */
    public void withdraw(double amount) throws Exception {
        if (amount > this.balance) {
            throw new Exception("Amount is greater than available balance.");
        }
        this.balance -= amount;
    }
```

Penn Engineering   Property of Penn Engineering |

38

BankAccount Class

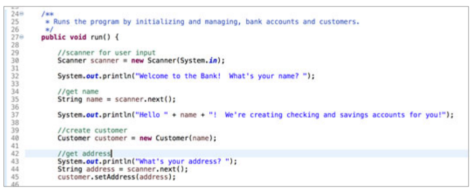


```java
    /**
     * Gets account type and balance.
     * @return String with all info
     */
    public String getAccountInfo() {
        return this.accountType + ": " + this.getBalance();
    }

    /**
     * Gets account balance.
     * @return rounded balance
     */
    public String getBalance() {
        DecimalFormat df = new DecimalFormat("#.##");
        df.setRoundingMode(RoundingMode.CEILING);

        return df.format(this.balance);
    }

    /**
     * Gets customer name and address.
     * @return String with all info
     */
    public String getCustomerInfo() {
        return this.customer.getName() + " from " + this.customer.getAddress();
    }
```

Penn Engineering   Property of Penn Engineering |

39

Bank Class



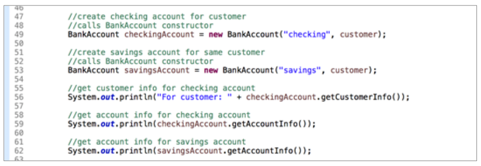
```java
    /**
     * Runs the program by initializing and managing, bank accounts and customers.
     */
    public void run() {

        //scanner for user input
        Scanner scanner = new Scanner(System.in);

        System.out.println("Welcome to the Bank!  What's your name? ");

        //get name
        String name = scanner.next();

        System.out.println("Hello " + name + "!  We're creating checking and savings accounts for you!");

        //create customer
        Customer customer = new Customer(name);

        //get address
        System.out.println("What's your address? ");
        String address = scanner.next();
        customer.setAddress(address);
```

PennEngineering

Property of Penn Engineering |

40

Bank Class



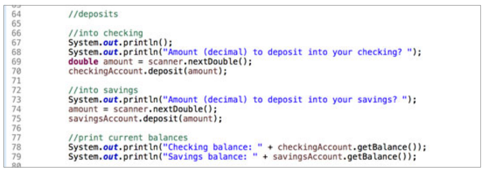
```java
        //create checking account for customer
        //calls BankAccount constructor
        BankAccount checkingAccount = new BankAccount("checking", customer);

        //create savings account for same customer
        //calls BankAccount constructor
        BankAccount savingsAccount = new BankAccount("savings", customer);

        //get customer info for checking account
        System.out.println("For customer: " + checkingAccount.getCustomerInfo());

        //get account info for checking account
        System.out.println(checkingAccount.getAccountInfo());

        //get account info for savings account
        System.out.println(savingsAccount.getAccountInfo());
```

PennEngineering

Property of Penn Engineering |

41

Bank Class

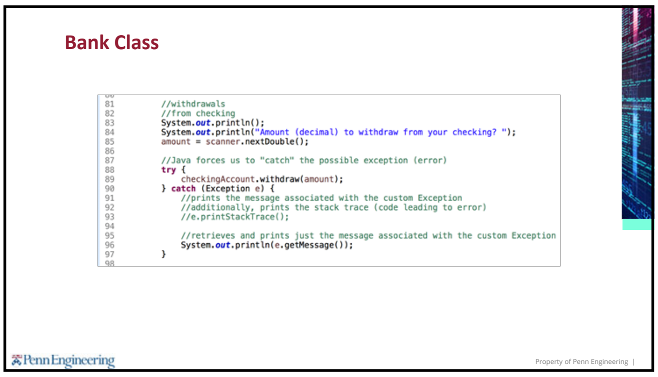


```java
        //deposits

        //into checking
        System.out.println();
        System.out.println("Amount (decimal) to deposit into your checking? ");
        double amount = scanner.nextDouble();
        checkingAccount.deposit(amount);

        //into savings
        System.out.println("Amount (decimal) to deposit into your savings? ");
        amount = scanner.nextDouble();
        savingsAccount.deposit(amount);

        //print current balances
        System.out.println("Checking balance: " + checkingAccount.getBalance());
        System.out.println("Savings balance: " + savingsAccount.getBalance());
```

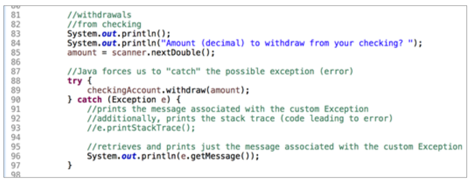
PennEngineering

Property of Penn Engineering |

42

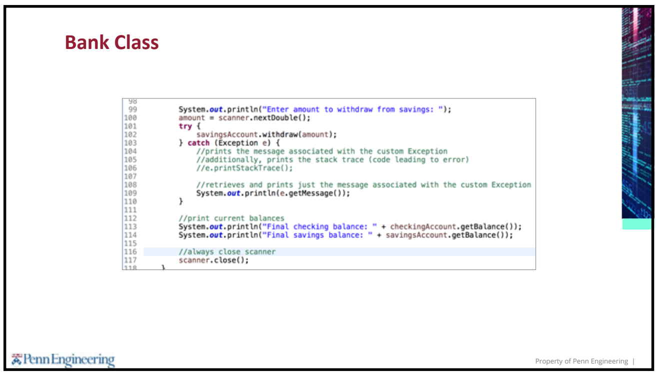


Bank Class

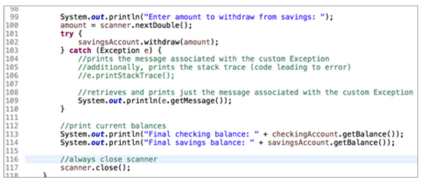


```java
        //withdrawals
        //from checking
        System.out.println();
        System.out.println("Amount (decimal) to withdraw from your checking? ");
        amount = scanner.nextDouble();

        //Java forces us to "catch" the possible exception (error)
        try {
            checkingAccount.withdraw(amount);
        } catch (Exception e) {
            //prints the message associated with the custom Exception
            //additionally, prints the stack trace (code leading to error)
            //e.printStackTrace();

            //retrieves and prints just the message associated with the custom Exception
            System.out.println(e.getMessage());
        }
```

43



Bank Class



```java
        System.out.println("Enter amount to withdraw from savings: ");
        amount = scanner.nextDouble();
        try {
            savingsAccount.withdraw(amount);
        } catch (Exception e) {
            //prints the message associated with the custom Exception
            //additionally, prints the stack trace (code leading to error)
            //e.printStackTrace();

            //retrieves and prints just the message associated with the custom Exception
            System.out.println(e.getMessage());
        }

        //print current balances
        System.out.println("Final checking balance: " + checkingAccount.getBalance());
        System.out.println("Final savings balance: " + savingsAccount.getBalance());

        //always close scanner
        scanner.close();
    }
```

44