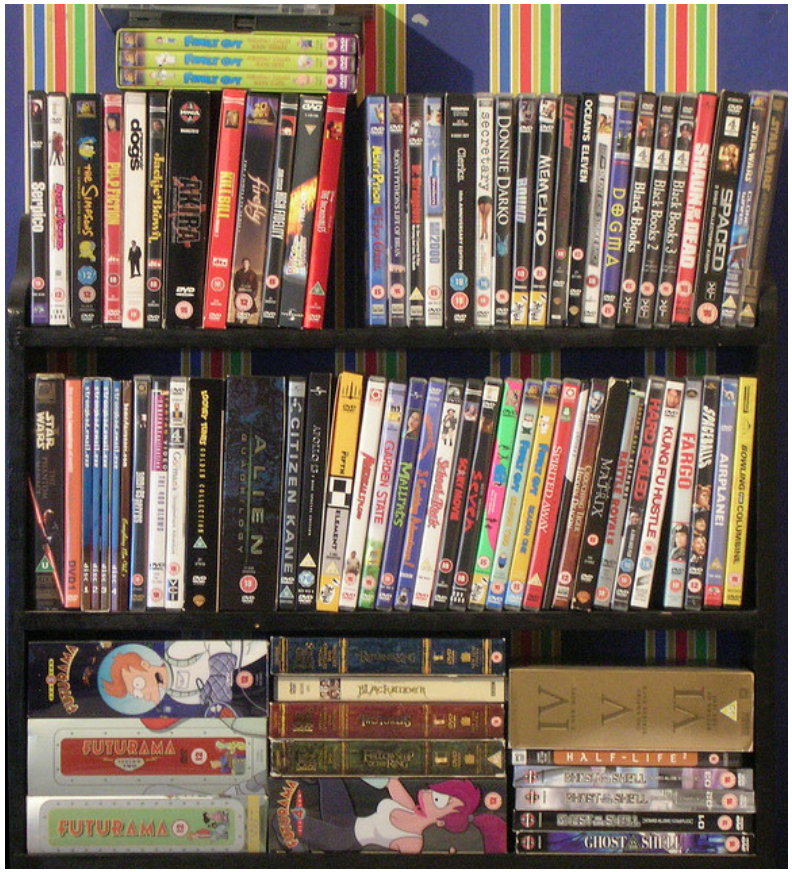
More Objects

Daniel Archambault

Previously in CS-115



Stephen Train Creative Commons

Instantiate those DVDs!

Previously in CS-115

- What is an abstract data type (ADT)?

Previously in CS-115

- What is an abstract data type (ADT)?
- What is the difference between an ADT and a data structure?

Previously in CS-115

- What is an abstract data type (ADT)?
- What is the difference between an ADT and a data structure?
- Why do we need ADTs in the first place?

Previously in CS-115

- What is an abstract data type (ADT)?
- What is the difference between an ADT and a data structure?
- Why do we need ADTs in the first place?
- What are the two essential parts of a class?

Previously in CS-115

- What is an abstract data type (ADT)?
- What is the difference between an ADT and a data structure?
- Why do we need ADTs in the first place?
- What are the two essential parts of a class?
- What is the difference between a class and an instance of a class?

Previously in CS-115

- What is an abstract data type (ADT)?
- What is the difference between an ADT and a data structure?
- Why do we need ADTs in the first place?
- What are the two essential parts of a class?
- What is the difference between a class and an instance of a class?
- What does private mean? What does public mean?

Previously in CS-115

- What is an abstract data type (ADT)?
- What is the difference between an ADT and a data structure?
- Why do we need ADTs in the first place?
- What are the two essential parts of a class?
- What is the difference between a class and an instance of a class?
- What does private mean? What does public mean?
- What is the difference between static and non-static?

Previously in CS-115

- But given a class, how do we use it properly?

Objects II

Reference Diagrams? Why are you doing this to me?!

- We are going over reference diagrams today.
- This is the most important lecture of CS-115 – especially overconfident students
- Reference diagrams help us understand
  - how data is stored by objects
  - computation on objects
  - data structures
- Understand things offline before you start programming.
- Student story... I don't know why he made us do that?

What are attributes?

- The data which defines an object
- When we create a new instance of a class
  - We get a copy of all non-static instance variables
  - Static variables have one variable back with the class
- Use constructors to initialize instance variables of a new object

```
 public class Book
{
   private String title;
   private String author;
   private int numPages;

    public Book (String title, String author, int numPages)
   {
      this.title = title;
      this.author = author;
      this.numPages = numPages;
   }
}

 public class BookTester
{
   ....
   Book b1 = new Book (“Swan Lake”, “Tchaikovsky”, 170);
   Book b2 = new Book (“Awake and Dreaming”, “Kit Pearson”, 140);
   ....
}
```

References are Critical

- Remember class types and simple types
  - Class types pass parameters by copying references
  - Simple types pass parameters by copying values
- last example no difference, but in this one, a big one!

Reference Example

```
public class Book
{
   private String title;
   private String author;
   private int numPages;

    public Book (String title, String author, int numPages)
   {
      this.title = title;
      this.author = author;
      this.numPages = numPages;
   }
}

public class BookTester
{
   ....
   String name = “Julie E. Czerneda”;
   Book b1 = new Book (“Hidden in Sight”, name, 500);
   Book b2 = new Book (“SI: Survival”, name, 401);
   ....
}
```

Calling Methods

- Static methods called on a class
  - Book.incCheckins ()
- Non-static methods called on a object
  - b1.setAuthor (“Me”)
  - this.setAuthor (“Me”)
- Although its allowed do not omit what comes before the “.”

Inside Methods

- Inside a static method you have access to:
  - static instance variables
- Inside a non-static method you have access to
  - static and non-static instance variables

Non-Static Methods

```
public class Book
{
    ...
     public String setAuthor (String author)
    {
        this.author = author;
    }
    ...
}

public class BookTester
{
    ...
    Book b1 = new Book (“Awake and Dreaming”, “Kit Pearson”, 140);
    (1) b1.setAuthor (“Vincent Thomas”);
    (2) Book.setAuthor (“Vincent Thomas”);

    ...
}
```

- Which is all right **(1)** or **(2)**

Static Methods Example

```
public class Book
{
   public static int totalNumberOfCheckins;
   ...
    public static void incCheckins()
   {
      totalNumberOfCheckins++;
   }
   ...
}

public class BookTester
{
   ...
   Book b1 = new Book ("Awake and Dreaming", "Kit Pearson", 140);
   (1) b1.incCheckins ();
   (2) Book.incCheckins ();

   ...
}
```

- Which is all right **(1)** or **(2)**

References and Methods

....
Book b1 = new Book (“Awake and Dreaming”, “Kit Pearson”, 140);
Book b2 = new Book (“Hidden in Sight”, “Julie E. Czerneda”, 500);
b1 = b2;
b1.setAuthor (“Jim John”);
....

- Don’t forget that variables of a class type contain references

Parameter Passing

- A value is copied into a local parameter for the function
- Once the function terminates, the value is destroyed
  - For simple types, it is the actual value stored in the variable
  - For class types, it is a reference to an object
- References outside the method scope do not change
- For variable declarations
  - simple type (small letter) type (`int, char`) value stored
  - class type, a reference to the object is stored

```
public class ParamTest {
   public static void primSwap (int x, int y)
  {
     int temp = x;
     x = y;
     y = temp;
  }

   public static void arraySwap (int x[], int y[])
  {
     int temp = x[0];
     x[0] = y[0];
     y[0] = temp;
  }

   public static void main (String args[])
  {
     int x = 6;
     int y = 7;
     int a[] = {1, 2, 3};
     int b[] = {4, 5, 6};
     ParamTest.primSwap (x, y);
     ParamTest.arraySwap (a, b);
  }
}
```

```
public class ParamTest {
  public static void primSwap (int x, int y)
  {
    int temp = x;
    x = y;
    y = temp;
  }

  public static void arraySwap (int x[], int y[])
  {
    int temp = x[0];
    x[0] = y[0];
    y[0] = temp;
    x = {7, 8, 6};
  }

  public static void main (String args[])
  {
    int x = 6;
    int y = 7;
    int a[] = {1, 2, 3};
    int b[] = {4, 5, 6};
    ParamTest.primSwap (x, y);
    ParamTest.arraySwap (a, b);
  }
}
```

```
public class BookModify {
   public static void bookChange (Book b1, Book b2)
  {
     b1.setAuthor ("Thomas William");
     b2.setTitle ("Simple Recipes");
  }

   public static void main (String args[])
  {
     Book b1 = new Book ("Timothy Taylor", "Stanley Park", 400);
     Book b2 = new Book ("Frank McCourt", "Angela's Ashes", 340);
     BookModify.bookChange (b1, b2);
  }
}
```

```
public class BookModify {
   public static void bookChange (Book b1, Book b2)
   {
      b1.setAuthor ("Thomas William");
      b2.setTitle ("Simple Recipes");
   }

   public static void main (String args[])
   {
      Book b1 = new Book ("Timothy Taylor", "Stanley Park", 400);
      BookModify.bookChange (b1, b1);
   }
}
```

```
public class BookModify {
   public void bookChange (Book b1, Book b2)
  {
     b1.setAuthor ("Thomas William");
     b2.setTitle ("Simple Recipes");
  }

   public static void main (String args[])
  {
     Book b1 = new Book ("Timothy Taylor", "Stanley Park", 400);
     BookModify.bookChange (b1, b1);
  }
}
```

- Does this compile?

```
public class BookModify {
   public static void bookChange (Book b1, Book b2)
   {
      b1.setAuthor ("Thomas William");
      b2 = new Book ("Dr. S.", "One Fish Two Fish", 14);
      b2.setTitle ("Simple Recipes");
   }

   public static void main (String args[])
   {
      Book b1 = new Book ("Timothy Taylor", "Stanley Park", 400);
      Book b2 = new Book ("Frank McCourt", "Angela's Ashes", 340);
      BookModify.bookChange (b1, b2);
   }
}
```

- Arrays of Class Types
  - arrays of references
  - they behave like references in parameter passing
  - you have an array of arrows to objects
- Initially, all the arrows point nowhere `null`

```
public class BookArrayModify {
   public static void bookArrayChange (Book[] bookArray)
  {
     bookArray[0].setAuthor ("Frank John");
     bookArray[2].setTitle ("Simple Recipes");
  }

   public static void main (String args[])
  {
     Book b1 = new Book ("Timothy Taylor", "Stanley Park", 400);
     Book b2 = new Book ("Dr. S.", "One Fish Two Fish", 14);
     Book b3 = new Book ("Frank McCourt", "Angela's Ashes", 340);
     Book[] bookList = new Book[3]; //understood null, null, null
     BookArrayModify.bookArrayChange (bookList);
  }
}
```

```
public class BookArrayModify {
   public static void bookArrayChange (Book[] bookArray)
  {
     bookArray[0].setAuthor ("Frank John");
     bookArray[2].setTitle ("Simple Recipes");
  }

   public static void main (String args[])
  {
     Book b1 = new Book ("Timothy Taylor", "Stanley Park", 400);
     Book b2 = new Book ("Dr. S.", "One Fish Two Fish", 14);
     Book b3 = new Book ("Frank McCourt", "Angela's Ashes", 340);
     Book[] bookList = {b1, b2, b3};
     BookArrayModify.bookArrayChange (bookList);
  }
}
```

```
public class BookArrayModify {
   public static void bookArrayChange (Book[] bookArray)
  {
     bookArray[0].setAuthor (“Frank John”);
     bookArray[2] = bookArray[1];
     bookArray[2].setTitle (“Simple Recipes”);
  }

   public static void main (String args[])
  {
     Book b1 = new Book (“Timothy Taylor”, “Stanley Park”, 400);
     Book b2 = new Book (“Dr. S.”, “One Fish Two Fish”, 14);
     Book b3 = new Book (“Frank McCourt”, “Angela’s Ashes”, 340);
     Book[] bookList = {b1, b2, b3}
     BookArrayModify.bookArrayChange (bookList);
  }
}
```