Accelerating Information Technology Innovation

http://aiti.mit.edu

India Summer 2012
Review Session – Java and Python

Intro to Java

Intro to Java

Make a class called `Animal`.

The constructor should take in the name of the animal and save it.

Intro to Java

```java
public class Animal {
    protected String mName;
    public Animal(String name) {
        this.mName = name;
    }
}
```

Intro to Java

Add a method that returns the name of the animal.

Intro to Java

```
public class Animal {
    /* ... */
    public String getName() {
        return this.mName;
    }
    /* ... */
}
```

Intro to Java

Make a new class `AnimalProgram` *with a main() method.*

In the main method, make an animal named "Divya".

Then make an animal named "Ian".

Get the name of the first animal you made, and print it.

Intro to Java

```java
public class AnimalProgram {
    public static void main(String[] args) {
        Animal divya = new Animal("Divya");
        Animal ian = new Animal("Ian");
        System.out.println(divya.getName());
    }
}
```

Intro to Java

Make a subclass of `Animal` *called* `Elephant`.

Make a second subclass of `Animal` *called* `Duck`.

Intro to Java

```
public class Elephant extends Animal {
    public Elephant(String name) {
        super(name);
    }
}

/* The Duck class will look similar. */
```

Intro to Java

Make a method called `speak` *in each class.*

Speak should print the name of the animal, and then print "speaks" if it is an `Animal`*, "trumpets" if it is an* `Elephant`*, and "quacks" if it is a* `Duck`*.*

Intro to Java

```java
public class Animal {
    /* ... */
    public void speak() {
        System.out.println(mName + " speaks");
    }
    /* ... */
}

/* The Elephant and Duck classes will look *
 * similar.                                 */
```

Intro to Java

Make an interface called `Flyable`*,*
which contains the method `fly`*.*
The method returns nothing.

Intro to Java

```
public interface Flyable {
    abstract public void fly();
}
```

Intro to Java

Make the `Duck` *class implement* `Flyable`*.*
When the `fly` *method in* `Duck` *is called,*
it should print the name of the animal,
and then print "flies".

Intro to Java

```java
public class Duck implements Flyable {
    /* ... */
    public void fly() {
        System.out.println(mName + " flies");
    }
    /* ... */
}
```

Intro to Java

Now make a class called `Airplane`.

`Airplane` *should also implement* `Flyable`.

Intro to Java

```java
public class Airplane implements Flyable {
    /* ... */
    public void fly() {
        System.out.println("A flying plane");
    }
    /* ... */
}
```

Intro to Java

Make an `ArrayList` *of 4* `Animals` *named "Alice", "Ben", "Chris", and "Dana".*

Print the name of the third animal in the array.

Intro to Java

```java
ArrayList<Animal> array =
  new ArrayList<Animal>();
array.add(new Animal("Alice"));
array.add(new Animal("Ben"));
array.add(new Animal("Chris"));
array.add(new Animal("Dana"));
System.out.println(array.get(2).getName());
```

Intro to Java

Make a `HashMap` *called* `myZoo` *that contains an* `Elephant` *named "Frank" and a* `Duck` *named "Georgia". The map should be keyed by the type of the animal.*

Now print the name of the elephant in the zoo.

Intro to Java

```java
HashMap<String, Animal> myZoo =
  new HashMap<String, Animal>();
myZoo.put("Elephant", new Elephant("Frank"));
myZoo.put("Duck", new Duck("Georgia"));
System.out.println(
  myZoo.get("Elephant").getName());
```

Intro to Java

In the main method, make a new string called `string1` *using the* `String` *constructor with the argument "Example".* *Make a second string called* `string2` *using the* `String` *constructor with the argument "Example".*

Print `string1 == string2`.
Print `string1.equals(string2)`
What do you notice?

Intro to Java

```java
String string1 = new String("Example");
String string2 = new String("Example");

// Prints "false"
System.out.println(string1 == string2);
// Prints "true"
System.out.println(string1.equals(string2));

/* .equals() method is used to compare *
 * object values, == compares pointers */
```

Break!

Intro to Python

Intro to Python

Make a class called `Animal`.

The constructor should take in the name of the animal and save it.

Intro to Python

```python
# NOTE: Indentation is important!
class Animal(object):
    def __init__(self, name):
        self.name = name
```

Intro to Python

Add a method that returns the name of the animal.

Intro to Python

```
class Animal(object):
    # ...
    def getName(self):
        return self.name

    # But accessing attributes directly
    # is better!
```

Intro to Python

In the main method, make an animal named "Divya".
Then make an animal named "Ian".
Get the name of the first animal you made, and print it.

Intro to Python

```
def main():
    divya = Animal("Divya")
    ian = Animal("Ian")
    print divya.name

if __name__ == "__main__":
    main()
```

Intro to Python

Make a subclass of `Animal` *called* `Elephant`.

Make a second subclass of `Animal` *called* `Duck`.

Intro to Python

```python
class Elephant(Animal):
    def __init__(self, name):
        Animal(self, name)

# The Duck class will look similar.
```

Intro to Python

Make a method called `speak` *in each class.*

Speak should print the name of the animal, and then print "speaks" if it is an `Animal`*, "trumpets" if it is an* `Elephant`*, and "quacks" if it is a* `Duck`*.*

Intro to Python

```
class Animal(object):
    # ...
    def speak(self):
        print self.name + " speaks"
```

Intro to Python

Make a `list` *of 4* `Animals`
named "Alice", "Ben", "Chris", and "Dana".

Print the name of the third animal in the list.

Add a fifth animal named "Eliza".

Intro to Python

```
myList = [Animal("Alice"), Animal("Ben"),
          Animal("Chris"), Animal("Dana")]
print myList[2].name
myList.append(Animal("Eliza"))
```

Intro to Python

Make a `dict` *called* `myZoo` *that contains an* `Elephant` *named "Frank" and a* `Duck` *named "Georgia". The dictionary should be keyed by the type of the animal.*

Now print the name of the elephant in the zoo.

Add an `Animal` *named "Hal" with the key "Tiger"*

Intro to Python

```
myZoo = {"Elephant": Elephant("Frank"),
         "Duck": Duck("Georgia")}
print myZoo["Elephant"].name
myZoo["Tiger"] = Animal("Hal")
```

Intro to Python

Make an `Animal` *called* `animal1`
whose name is "Ben".
Make a second `Animal` *called* `animal2`
whose name is also "Ben".

Print `animal1 == animal2`.

Intro to Python

```python
animal1 = Animal("Ben")
animal2 = Animal("Ben")
# Prints "False"
print animal1 == animal2
```

Intro to Python

In the `Animal` *class, create a method called* `__eq__`.
This method should take one argument,
which will be an `Animal` *object.*
If the two `Animals` *have the same name, return* `True`.
Otherwise, return `False`.

Print `animal1 == animal2`.
What do you notice?

Intro to Python

```
class Animal(object):
    # ...
    def __eq__(self, other):
        return self.name == other.name

# Prints "True"
print animal1 == animal2

# __eq__() overloads the == operator
# Python can overload other operators
```