The \texttt{const} Keyword

Extreme Encapsulation
Humble Beginnings

• There are often cases in coding where it is helpful to use a `const` variable in a method or program.
  – Even when working with fixed values, it is best to abstract them with variable names.
  – It’s more helpful (debugging-wise) to see “array_length” than tons of copies of the same number everywhere.
With object-orientation, many classes may take permanent values which are unique to each instance of the class, specified during initialization.

- As these should not change at any point in the object’s lifetime, `const` makes sense.
Humble Beginnings

• The use of `const` is fairly straightforward for the primitive data types – the basic building blocks of the language.

• Things get more complicated when we use `const` with pointers and with objects.
const and Objects

• What would it mean for an object to be const?
**const** and Objects

• What would it mean for an object to be **const**?
  – If declared **const**, an object should not be modifiable.
  – Problem: how can we use its methods while being sure not to modify it?
**const and Objects**

- In C++, whenever a variable is declared `const`, no modifications are allowed to it, *in a by-value manner*.
  - As the compiler is not powerful enough to ensure that its methods do not modify it, by default C++ blocks all use of *any* class methods.
  - This would be a **huge** problem for encapsulation.
**const and Objects**

- The C++ solution to the problem this poses: *functions* can be declared **const**.
  - Appending the **const** keyword to a function signifies that the method is not allowed to alter the class in any manner.
  - Inside that method, *all* fields of the class will be treated as if they were declared **const**.
Let us now examine how this would look in code, through our frequent Person class example.
public class Person
{

    private:
    const string name;
    int age;

    public:
    Person(string name, int age)
    string getName() const;
    int getAge() const;
    void haveABirthday();

}
string Person::getName() const
{
    return this->name;
}

int Person::getAge() const
{
    return this->age;
}
public void haveABirthday()
{
    this->age++;
}

• Note: declaring this method as const would result in a compile-time error, as age would be treated as const within the method.
Which of the following code lines is invalid?

```cpp
const Person p(“Harrison Ford”, 75);

string name = p.getName();
int age = p.getAge();
p.haveABirthday();
```
const and Objects

• Which of the following code lines is invalid?

```cpp
p.haveABirthday();
```

• As this method is not declared `const`, a compile-time error would result from this method being called upon `const p`. 