7.5 Mutators, accessors, and private helpers

Mutators and accessors

A class' public functions are commonly classified as either mutators or accessors.

- A ***mutator*** function may modify ("mutate") a class' data members.
- An ***accessor*** function accesses data members but does not modify a class' data members.

Commonly, a data member has two associated functions: a mutator for setting the value, and an accessor for getting the value, known as a ***setter*** and ***getter*** function, respectively, and typically with names starting with set or get. Other mutators and accessors may exist that aren't associated with just one data member, such as the Print() function below.

Accessor functions usually are defined as const to make clear that data members won't be changed. The keyword ***const*** after a member function's name and parameters causes a compiler error if the function modifies a data member. If a const member function calls another member function, that function must also be const.

Figure 7.5.1: Mutator and accessor public member functions.

```
#include <iostream>
#include <string>
using namespace std;

class Restaurant {
   public:
      void   SetName(string restaurantName); // Mutator
      void   SetRating(int userRating);      // Mutator
      string GetName() const;                // Accessor
      int    GetRating() const;              // Accessor
      void   Print() const;                  // Accessor

   private:
      string name;
      int rating;
};

void Restaurant::SetName(string restaurantName) {
   name = restaurantName;
}

void Restaurant::SetRating(int userRating) {
   rating = userRating;
}

string Restaurant::GetName() const {
   return name;
}

int Restaurant::GetRating() const {
   return rating;
}

void Restaurant::Print() const {
   cout << name << " -- " << rating << endl;
}

int main() {
   Restaurant myPlace;

   myPlace.SetName("Maria's Diner");
   myPlace.SetRating(5);

   cout << myPlace.GetName() << " is rated ";
   cout << myPlace.GetRating() << endl;

   return 0;
}
```

```
Maria's Diner is rated 5
```

Feedback?

PARTICIPATION ACTIVITY 7.5.1: Mutators and accessors.

1) A mutator should not change a class' private data members.

- True
- False

Correct

A mutator "mutates", meaning changes, a class' private data members.

False

2) An accessor should not change a class' private data members.
   - True
   - False

Correct

An accessor accesses private data members, and then returns some value or carries out some action. An accessor should not change private data members, and thus is usually defined as const.

3) A private data member sometimes has a pair of associated set and get functions.
   - True
   - False

Correct

A pair of associated set and get functions is sometimes, but not always, the case, and depends on whether the data member should be known to the class user. Some private data members are completely hidden from the user.

4) Accessor functions are required to be defined as const.
   - True
   - False

Correct

Accessor functions *should* usually be defined as const, so that the compiler will report an error if the function tries to modify a data member, but const is not required.

5) A const accessor function may call a non-const member function.
   - True
   - False

Correct

The compiler will not allow such a call. A const accessor function should not change private data, either itself or via calls to other functions.

Feedback?

Private helper functions

A programmer commonly creates private functions, known as ***private helper functions***, to help public functions carry out tasks.

PARTICIPATION ACTIVITY 7.5.2: Private helper member functions.

Start 2x speed

```
class MyClass {
   public:
      void Fct1();
   private:
      int numA;
```

```
int main() {
   MyClass SomeObj;

   SomeObj.Fct1();
```

OK

```
    int FctX();
};
void MyClass::Fct1() {
   numA = FctX();
   ...
}

int MyClass::FctX() {
   ...
}
```

OK

```
   ...
   SomeObj.FctX();
   return 0;
}
```

Error

Feedback?

PARTICIPATION ACTIVITY 7.5.3: Private helper functions.

1) A class' private helper function can be called from main().
   - ○ True
   - ◉ False

Correct

A class' private functions can only be called from the class' other functions.

2) A private helper function typically helps public functions carry out their tasks.
   - ◉ True
   - ○ False

Correct

Private helper functions can lead to cleaner public function definitions.

3) A private helper function cannot call another private helper function.
   - ○ True
   - ◉ False

Correct

No such restriction exists. A private helper can be called from any other function of the class, whether public or private.

4) A public member function may not call another public member function.
   - ○ True
   - ◉ False

Correct

No such restriction exists. Such a call can reduce redundant code.

Feedback?

CHALLENGE ACTIVITY 7.5.1: Mutators, accessors, and private helpers.

Jump to level 1

Type the program's output.

```
Human
```

```
#include <iostream>
using namespace std;

class Dog {
   public:
      void SetWeightAndAge(int weightToSet, int ageToSet);
      int GetHumanYears() const;
   private:
      int years;
      int weight;
      string size;
      int humanYears;
      void SetHumanYears();
};

void Dog::SetWeightAndAge(int weightToSet, int yearsToSet) {
   weight = weightToSet;

   if (weight <= 20) {
      size = "small";
   }
   else if (weight <= 45) {
      size = "medium";
   }
   else {
      size = "large";
   }

   years = yearsToSet;
   SetHumanYears();
}

void Dog::SetHumanYears() {
   int factor;

   if (size == "small") {
      factor = 6;
   }
   else if (size == "medium") {
      factor = 7;
   }
   else {
      factor = 8;
   }

   humanYears = years * factor;
}

int Dog::GetHumanYears() const {
   return humanYears;
}

int main() {
   Dog buddy;

   buddy.SetWeightAndAge(18, 2);

   cout << "Human years: " << buddy.GetHumanYears();
   return 0;
}
```

1 2 3

Check Next **Done**. Click any level to practice more. Completion is preserv

The SetWeightAndAge mutator calls the SetHumanYears private helper. The human years depends on the dog's size.

Yours	`Human years: 12`
Expected	`Human years: 12`

Feedback?