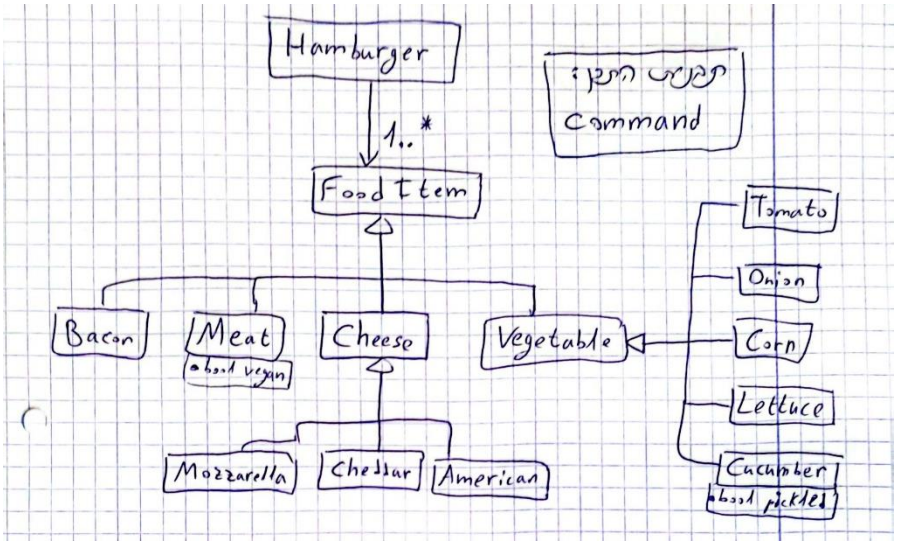
א. מימוש:

```
class Hamburger {
    vector<shared_ptr<FoodItem>> m_contents;
    bool m_upper;
    int m_taste;

    bool is_Meat(const shared_ptr<FoodItem>& item) const {
        return dynamic_cast<Meat*>(item.get()) != nullptr;
    }

    bool has_Meat() const {
        for(const shared_ptr<FoodItem>& item : m_contents) {
            if(is_Meat(item)){
                return true;
            }
        }
        return false;
    }

public:
    Hamburger(const list<shared_ptr<FoodItem>>& contents, bool upper) :
      m_contents(), m_upper(upper), m_taste(1)
    {
        for(const shared_ptr<FoodItem>& item : contents) {
            addContent(item);
        }
        if(!has_Meat()) {
            throw MeatlessBurger("A Burger has to have meat");
        }
    }

    void addContent(const shared_ptr<FoodItem>& item) {
        m_contents.push_back(item->clone());
        if(!is_Meat(item) || has_Meat()) {
            m_taste += item->Taste();
        }
    }

    int Taste() const {
        return m_taste;
    }

    float Quality() const {
        return m_taste/m_contents.size();
    }
```

```
    bool addUpperBread() {
        if(m_upper){
            return false;
        }
        m_upper = true;
        return true;
    }

    bool removeUpperBread() {
        if(!m_upper){
            return false;
        }
        m_upper = false;
        return true;
    }
};

class MeatlessBurger : public runtime_error {
public:
    MeatlessBurger(const string& err_msg) : runtime_error(err_msg) {}
};
```

ב. תרשים:



ג. מימוש:

```
bool operator<(const Hamburger& h1, const Hamburger& h2) {
    return h1.Quality() < h2.Quality();
}

void sortByQuality (vector<Hamburger>& hamburgers) {
    sort(hamburgers.begin(), hamburgers.end());
}
```

ד. מימוש:

```
class Meat : public FoodItem {
    int m_weight;
    static const int TASTE = 10;

public:
    Meat(int weight) :
        FoodItem(TASTE), m_weight(weight)
    {}
    ~Meat() override = default;

    shared_ptr<FoodItem> clone() const override {
        return shared_ptr<FoodItem>(new Meat(m_weight));
    }
};

class Cheese : public FoodItem {
public:
    Cheese(int taste) : FoodItem(taste) {}
    virtual ~Cheese() override = default;
};

class Cheddar : public Cheese {
    static const int TASTE = 9;

public:
    Cheddar() : Cheese(TASTE) {}
    ~Cheddar() override = default;

    shared_ptr<FoodItem> clone() const override {
        return shared_ptr<FoodItem>(new Cheddar());
    }
};

class Mozzarella : public Cheese {
    static const int TASTE = 8;

public:
    Mozzarella() : Cheese(TASTE) {}
    ~Mozzarella() override = default;

    shared_ptr<FoodItem> clone() const override {
        return shared_ptr<FoodItem>(new Mozzarella());
    }
};
```

```
class American : public Cheese {
    static const int TASTE = 7;

public:
    American() : Cheese(TASTE) {}
    ~American() override = default;

    shared_ptr<FoodItem> clone() const override {
        return shared_ptr<FoodItem>(new American());
    }
};
```

ה. תרשים: